
AMSTRAD
6128
plus
LE LIVRE
DU BASIC
EDITIONS MICRO APPLICATION
MA
LIVRE DATA BECKER

Kowal

Le grand livre du

BASIC

EDITIONS MICRO APPLICATION

Auteur Kowal

Traducteur Pascal Haussman

ISBN : 2-86899-421-0

Collection dirigée par Philippe Olivier
Edition réalisée par Frédérique Beaudonnet

Préface

L'AMSTRAD 6128+ est certainement un ordinateur extrêmement puissant qui ne craint pas la comparaison. Il peut être utilisé aussi bien pour des applications privées que pour des applications professionnelles. Grâce notamment à son logiciel système très polyvalent, il se prête parfaitement à des applications de type professionnel. C'est ainsi que la dernière version de CP/M (CP/M Plus) constitue sans aucun doute un système d'exploitation intéressant pour les ordinateurs 8 bits parmi ceux qui existent actuellement. CP/M Plus se distingue notamment par un nombre considérable de textes HELP très complets qui sont particulièrement intéressants pour ceux qui n'ont pas encore l'habitude du travail sous CP/M. Mais CP/M Plus se distingue également par une gestion de fichiers améliorée. Par ailleurs, CP/M Plus est compatible avec la version précédente, CP/M 2.2, de sorte que les programmes innombrables qui existent sur le marché pour CP/M tournent également sous le nouveau système d'exploitation.

Dans le présent ouvrage, nous ne décrirons cependant pas plus avant le monde de CP/M. Cet ouvrage s'adresse plutôt à ceux des amateurs d'informatique qui souhaitent travailler principalement sous AMSDOS et en BASIC AMSTRAD. Nous supposons par ailleurs que le lecteur dispose de certaines connaissances de base dans le langage de programmation BASIC mais il est cependant également possible de comprendre cet ouvrage après avoir étudié attentivement le manuel du BASIC fourni avec la machine ainsi que les chapitres d'introduction du présent ouvrage.

Il ne s'agit pas cependant ici d'un livre pour 'débuter' mais plutôt d'un livre pour les 'débutants avisés'. C'est ainsi par exemple que nous ne réexpliquons pas dans le détail toutes les instructions utilisées mais que nous cherchons plutôt à montrer les relations établies entre des instructions ou des groupes d'instructions de types très différents. Nous y ajoutons une quantité de trucs et astuces ainsi que quelques idées et applications de programmes que nous espérons intéressantes pour les amateurs de programmation.

Il va de soi que nous avons cherché en premier lieu à illustrer les particularités et les caractéristiques techniques du 6128 +.

Le lecteur tirera le profit maximum de cet ouvrage s'il modifie les exemples ou programmes présentés ou s'il les adapte à ses propres besoins. La manière la plus simple d'apprendre à programmer consiste à faire des essais avec les instructions et les programmes. Les exercices "théoriques" débouchent beaucoup plus rarement sur des succès. Et s'il vous arrive malgré tout de rencontrer des problèmes qui vous semblent insolubles, alors n'oubliez pas que c'est en forgeant qu'on devient forgeron.

Si vous disposez de logiciels pour le CPC 464/664, vous pouvez les faire tourner sans problème sur le 6128. Cela ne vaut cependant que pour les programmes BASIC. Pour les programmes écrits en langage machine, un travail d'adaptation non négligeable est indispensable. Il n'y a donc pas de compatibilité complète des systèmes CPC.

Sommaire

1. L'ordinateur et son langage

1.1. L'Amstrad 6128 +

Comparé à l'AMSTRAD CPC 464/664 et à d'autres modèles courants d'ordinateurs familiaux, les 128 K octets de l'AMSTRAD 6128 + sont ce qu'il y a de véritablement nouveau et de remarquable. Les ordinateurs 8 bits comme le CPC 6128 ne peuvent normalement adresser que 64 K octets mais AMSTRAD offre avec le procédé appelé bank switching une solution vraiment intéressante pour l'utilisation du second bloc de 64 K octets. Ce second bloc de 64 K octets est en effet appelé avec des instructions RSX très simples (RSX signifie Resident System eXtension). Ces instructions RSX constituent une extension du jeu d'instructions. Vous obtenez ces instructions supplémentaires en lançant le programme BANKMAN.BAS qui se trouve sur la première face de la disquette système. Ces instructions sur lesquelles nous reviendrons plus en détail dans le cours de ce livre vous permettent d'employer, au choix, ce second bloc de 64 K octets soit comme disque RAM soit comme mémoire pour stocker des pages d'écran entières.

Nous vous expliquons en détail, au chapitre 9.3, quelles pseudo-opérations de fichier peuvent être exécutées lorsqu'on utilise le second bloc de 64 K de RAM comme disque RAM. L'emploi de la RAM supplémentaire pour des opérations écran est décrit dans la partie graphique de cet ouvrage (chapitres 6.11 et 6.12). Vous trouverez également dans les autres chapitres d'autres indications pour l'utilisation des 128 K octets en BASIC.

Mais même si on oublie ses 128 K, le 6128 + reste un super-ordinateur. Il possède un certain nombre de propriétés remarquables et pratiques qui sont largement inégalées.

C'est ainsi que le 6128 + est doté d'un jeu d'instructions BASIC très complet qui n'a pas à rougir, même comparé à ce qu'on appelle le standard MSX. Il possède en plus des instructions

standard qu'on trouve sur des ordinateurs plus primitifs des extensions nombreuses parmi lesquelles nous voudrions notamment souligner :

- La définition de fenêtres (windows) sur l'écran, chaque fenêtre se comportant exactement comme une petit écran autonome et pouvant être appelée avec les instructions d'entrée et de sortie habituelles.
- La commande d'interruptions à l'expiration de délais mesurés par une horloge en temps réel.

Mais dans le développement du 6128 +, on a également attribué une grande importance aux propriétés graphiques et aux couleurs. C'est ainsi que vous disposez des possibilités suivantes :

❶ Suivant le mode sélectionné :

- Résolution graphique pouvant atteindre 640 points sur 200
- Jusqu'à 80 caractères de texte par ligne
- 27 couleurs différentes
- 16 couleurs maximum pouvant être représentées imultanément

❷ Des instructions graphiques puissantes

❸ Des caractères graphiques

❶ Des caractères définis par l'utilisateur

Les possibilités sonores du 6128 + constituent un autre atout important. C'est ainsi qu'il est possible de produire simultanément trois sons différents avec une hauteur de note pouvant varier sur 8 octaves. Il est en outre possible de sortir la musique produite par l'ordinateur sur une chaîne HIFI connectée au système.

Comme chaque canal de son dispose d'une file d'attente pouvant comporter jusqu'à un maximum de 5 instructions

sonores, il est possible de produire des sons indépendamment du déroulement du programme.

Mais même en ne tenant pas compte du second bloc de 64 K octets de RAM, vous disposez d'une place mémoire importante. Vous pouvez d'ailleurs aisément le constater par vous-même en entrant :

```
?FRE(0)
```

Immédiatement après avoir allumé votre ordinateur. La fonction FRE vous fournit en effet la taille de la place mémoire non encore utilisée par le BASIC.

Immédiatement après la mise sous tension de l'ordinateur, vous obtenez une valeur de tout de même 42249 octets (1 K octets = 1024 octets). Cela représente certainement une valeur suffisante pour la plupart des applications élémentaires.

Lorsque vous éteignez votre CPC, tout ce qui se trouvait dans la mémoire de travail est bien sûr irrémédiablement perdu. Heureusement, il existe évidemment des mémoires externes pour conserver vos programmes et fixer vos données. Et pour cela vous n'avez pas besoin d'une machine supplémentaire puisque vous pouvez employer le lecteur de disquette intégré.

Dans sa version de base, votre AMSTRAD 6128 + constitue donc déjà un système informatique complet, prêt à fonctionner. Il vous permet d'exécuter les opérations de base du traitement des données :

- Réception des données ou informations (à travers le clavier).
- Stockage (interne ou externe) des données ou informations.
- Evaluation ou traitement de ces données.
- Sortie de résultats (sur l'écran).

La sortie des résultats ne se fait cependant pas obligatoirement sur l'écran.

Si vous connectez au système une imprimante ou une machine à écrire électronique moderne, vous pouvez faire sortir noir sur blanc les résultats de vos programmes ou bien les listings de programmes (hardcopy). Un certain nombre des tâches effectuées par les ordinateurs seraient d'ailleurs impensables s'il n'était pas possible de connecter au système des périphériques de ce type (songez par exemple au traitement de texte).

Le 6128 + est doté, pour la connexion de l'imprimante, d'une interface parallèle normale de sorte que vous disposez à cet égard d'un très grand choix d'imprimantes.

Mais vous pouvez aussi utiliser cette interface pour transférer des données vers d'autres ordinateurs. Il n'est en effet généralement pas possible d'utiliser des cassettes ou des disquettes pour ce type de transferts de données car les formats d'écriture sont presque toujours différents.

Si vous voulez donc relier votre 6128 + à un autre ordinateur, il vous suffit de disposer d'un câble reliant la connexion imprimante au port utilisateur de l'autre ordinateur lequel doit bien sûr également disposer d'une interface parallèle comme c'est par exemple le cas des ordinateurs Commodore. Il vous faut cependant encore un programme machine pour l'ordinateur récepteur pour recevoir les données venant du 6128 + et pour les convertir dans le code de la machine réceptrice. En indiquant alors 8 comme numéro de canal, vous pourrez utiliser des instructions telles que PRINT ou LIST pour le transfert de données ou de programmes.

Nous n'évoquerons cependant pas plus en détail cette possibilité du transfert de données pour ne pas sortir du cadre de cet ouvrage. D'ailleurs, une application de ce type ne devrait pas en principe présenter de grand intérêt pour des programmeurs qui ne sont pas encore très confirmés. De même, nous n'avons pas évoqué dans la suite de cet ouvrage l'emploi d'un coupleur acoustique ou d'un modem. La plupart des chapitres sont bien plutôt consacrés à la configuration de base du système, c'est-à-dire à l'AMSTRAD 6128 + avec le lecteur de disquette intégré et le moniteur AMSTRAD monochrone.

Bien sûr, quelques applications telles que le traitement de texte ne pourront être réalisées sans imprimante. Mais, dans ce cas, nous vous l'indiquerons chaque fois que ce sera nécessaire.

La possibilité d'employer différentes couleurs n'a de même été indiquée que lorsqu'elle présentait un intérêt réel ou lorsqu'elle était même nécessaire. Ceux qui ne possèdent qu'un moniteur monochrome vert ne se sentiront donc pas frustrés outre mesure. Il n'est d'ailleurs pas indispensable que chaque sortie de résultat sur l'écran emploie autant de couleurs que possible. Nous avons donc préféré montrer l'emploi des couleurs par un exemple (voir le chapitre 6) de façon à ce que le lecteur intéressé puisse éventuellement transformer en fonction de ses goûts tout ce qui apparaît en noir et blanc dans nos programmes.

1.2. Organiser son travail en Basic

Le langage de programmation BASIC fait partie de ce qu'on appelle les langages orientés utilisateur. Au contraire de ce qu'on appelle les langages orientés machine (Assembleur, etc...), il n'est pas nécessaire ici de se charger de l'adressage de la mémoire. Cette tâche est accomplie par le système d'exploitation de l'ordinateur.

Le BASIC convient parfaitement aux débutants programmeurs car du fait de sa structure simple il facilite la démarche de la programmation et permet aisément le passage à d'autres langages de programmation :

```
BASIC = Beginners All purpose Symbolic Instruction Code
```

Le BASIC se compose, comme d'autres langages de programmation, de commandes, d'instructions, de fonctions et d'opérateurs. On entend ici par commandes des ordres qui s'adressent directement au système d'exploitation de l'ordinateur. Les commandes indiquent de façon presque immédiate à l'ordinateur ce qu'il doit faire. Les instructions sont par contre des instructions de programmes qui ne seront traitées que lors du déroulement du programme. Elles ne prennent donc effet qu'après le lancement du programme.

Les fonctions sont des instructions qui calculent la valeur d'une expression. Il s'agit très souvent de notions mathématiques telles que sinus, cosinus ou logarithme. Les opérateurs, enfin, réalisent des opérations logiques. Les opérateurs sont surtout utilisés pour la programmation des conditions (instructions IF).

Toutefois, la distinction opérée ici entre commandes et instructions ne s'applique pas au BASIC AMSTRAD. En effet, en BASIC AMSTRAD, toutes les instructions peuvent également être employées comme des commandes et toutes les commandes aussi comme des instructions. Cette distinction traditionnelle a cependant une signification pédagogique. Elle permet en effet de distinguer des instructions qui sont utilisées principalement comme des commandes. Ces instructions servent en général à l'organisation du travail sur l'ordinateur.

Voici maintenant une liste des principales tâches d'organisation avec les commandes correspondantes :

Lancement d'un programme	RUN
Chargement d'un programme à partir de la disquette	LOAD
Ecriture d'un programme sur disquette	SAVE
Sortie de toutes les lignes de programme sur l'écran	LIST
Vidage de l'écran	CLS
Correction d'une ligne de programme	EDIT
Suppression d'un programme figurant en mémoire	NEW
Suppression d'une section de programme	DELETE
Numérotation automatique des lignes	AUTO
Reprise de l'exécution d'un programme interrompu	CONT
Sortie du catalogue d'une disquette	CAT
Renumérotation d'un programme	RENUM
Combinaison de deux programmes	CHAIN MERGE, MERGE

Nous vous prions de vous reporter à votre manuel d'utilisation pour connaître les différents paramètres de ces commandes. Les exemples qui vous y sont donnés pour ces instructions sont aisés à reproduire de sorte qu'il n'est pas nécessaire de les répéter ici. Vous trouverez également dans votre manuel une puissante solution de rechange pour EDIT. Il s'agit de la méthode avec le curseur COPY. Cette méthode vous permet de réaliser très rapidement des corrections. Ici non plus nous

n'avons rien à ajouter à la description détaillée que vous trouvez dans votre manuel d'utilisation.

- **Mais voyons maintenant comment un programme BASIC se présente.**

Tout programme BASIC se compose d'une suite d'instructions du programme qui sont entrées ligne par ligne dans l'ordinateur. Les lignes de programme sont traitées par l'ordinateur l'une après l'autre lorsque le programme est lancé. Chaque ligne reçoit pour cela un numéro de ligne permettant de définir l'ordre des instructions dans le programme. Si l'on fait abstraction de la possibilité de branchements dans un programme, le principe est que la ligne portant le numéro le plus petit sera toujours traitée avant la ligne de numéro plus élevé.

Votre ordinateur reconnaît la fin d'une ligne par le fait que l'utilisateur ait appuyé à cet endroit sur la touche RETURN ou ENTER. Une ligne peut aussi comprendre plusieurs instructions. Pour que l'ordinateur sache où une instruction se termine et où commence la suivante, il faut placer un double point entre deux instructions consécutives.

Lorsque vous allumez votre 6128 +, il se présente avec le message READY. Il est alors prêt à recevoir des commandes. Si vous voulez maintenant qu'il résolve par exemple un problème de multiplication élémentaire, il est inutile d'écrire un programme BASIC. Vous pouvez en effet entrer des commandes en mode direct, c'est-à-dire sans les faire précéder de numéros de lignes. Ces commandes seront alors exécutées immédiatement. Ce mode de fonctionnement est surtout employé lorsqu'on utilise l'ordinateur comme une 'calculatrice de poche'. Le problème 256 * 317 pourrait donc être résolu avec l'entrée suivante :

```
? 256*317 (ENTER)
```

Le point d'interrogation, que nous avons déjà utilisé au chapitre précédent pour connaître avec FRE la place mémoire disponible, fait ici office d'abréviation de l'instruction PRINT.

Cette instruction est une des instructions BASIC les plus importantes car elle est employée pour la sortie de résultats ou, comme dans l'exemple ci-dessus, pour le calcul.

1.3. Comment naît un programme

Avant que vous ne transformiez une idée ou un problème en une application programmée, nous vous conseillons de toujours essayer de réfléchir à la méthode que vous comptez employer pour la solution de votre problème. La conception d'une méthode logique de solution du problème doit être considérée comme une condition nécessaire à la réalisation d'un programme. Il est pratiquement inutile de s'asseoir devant son ordinateur si on n'a pas auparavant fourni ce travail de réflexion. Cela s'applique notamment aux débutants.

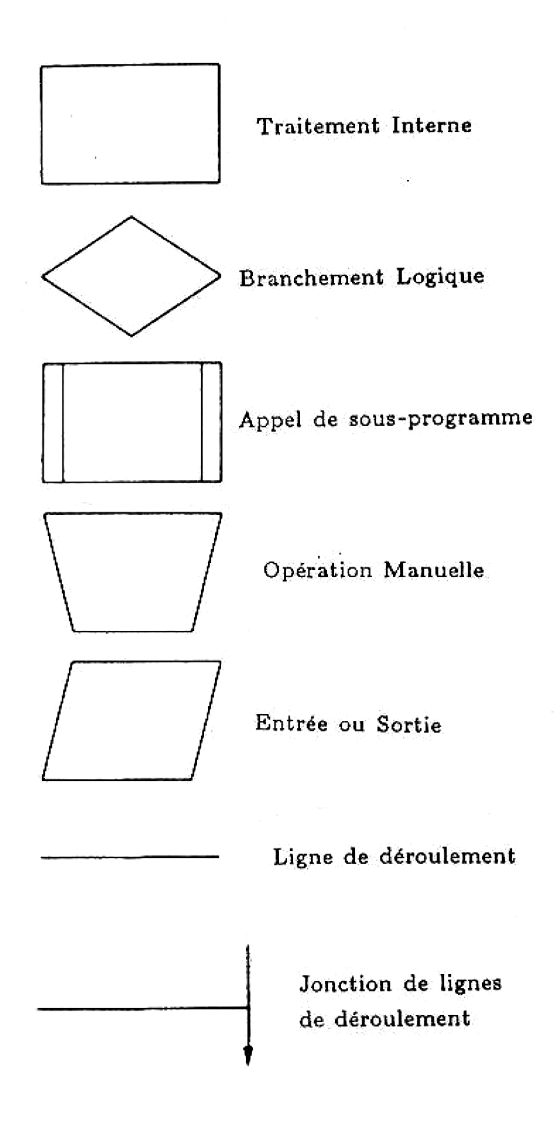
Mais même le programmeur expérimenté, s'il veut écrire un 'bon' programme, devra prendre sur lui de fournir ce travail de préparation car cela lui évitera de perdre un temps considérable lors des tests de mise au point du programme. Ce travail préalable permet en outre de bâtir des programmes ayant une structure claire et dont la logique pourra donc également être compréhensible par des personnes autres que le programmeur lui-même.

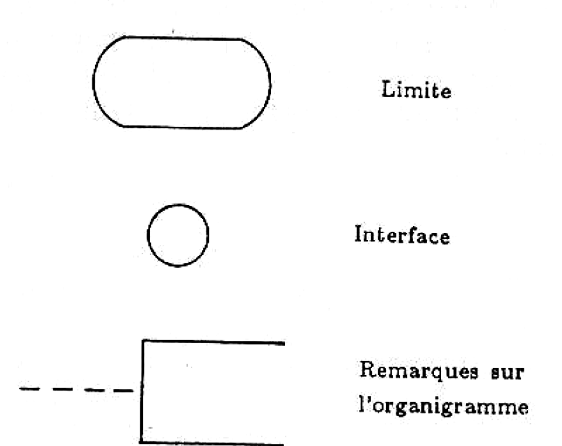
Mais en quoi consiste donc le travail de préparation de la programmation proprement dite ?

Pour résoudre un problème déterminé de traitement électronique des données, il est d'abord nécessaire d'analyser et de préciser le problème posé. Si vous voulez par exemple calculer une moyenne pour une série de mesures relevées, il vous faut savoir la formule que vous avez l'intention d'employer. L'une des formules possibles est la moyenne arithmétique qui consiste à diviser la somme de toutes les valeurs par le nombre de ces valeurs.

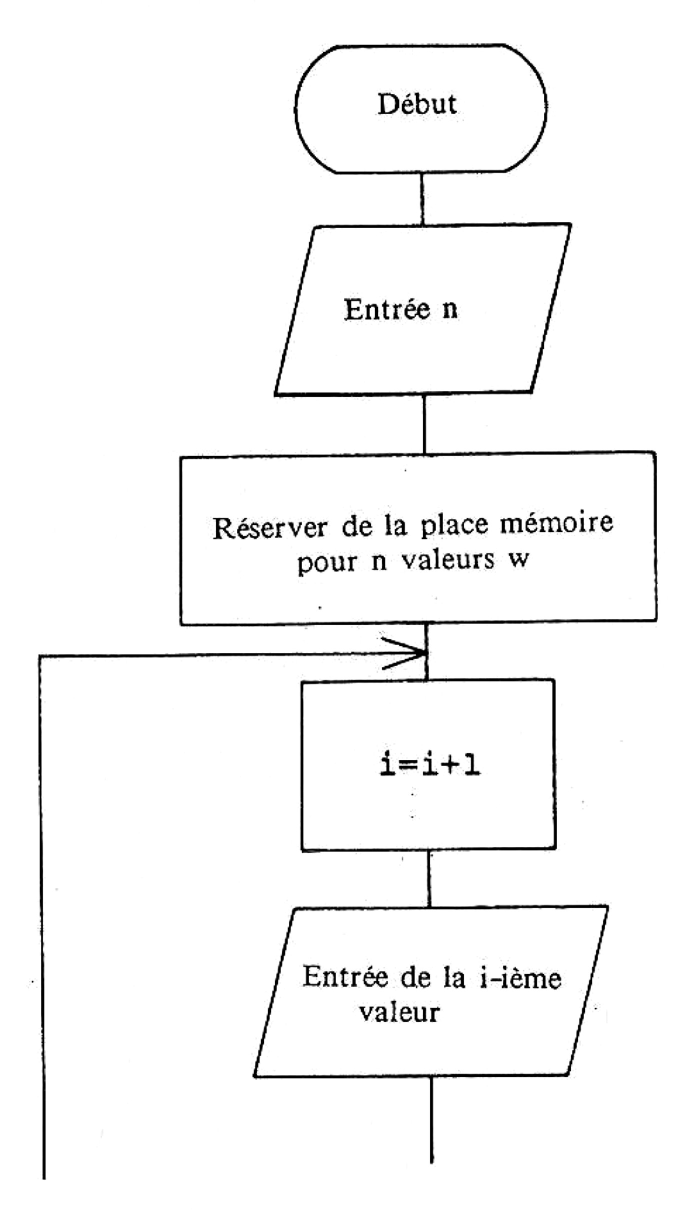
L'étape suivante est alors d'une importance capitale: il s'agit en effet d'esquisser la séquence de différentes tâches qui permettra de résoudre le problème. On représente généralement les

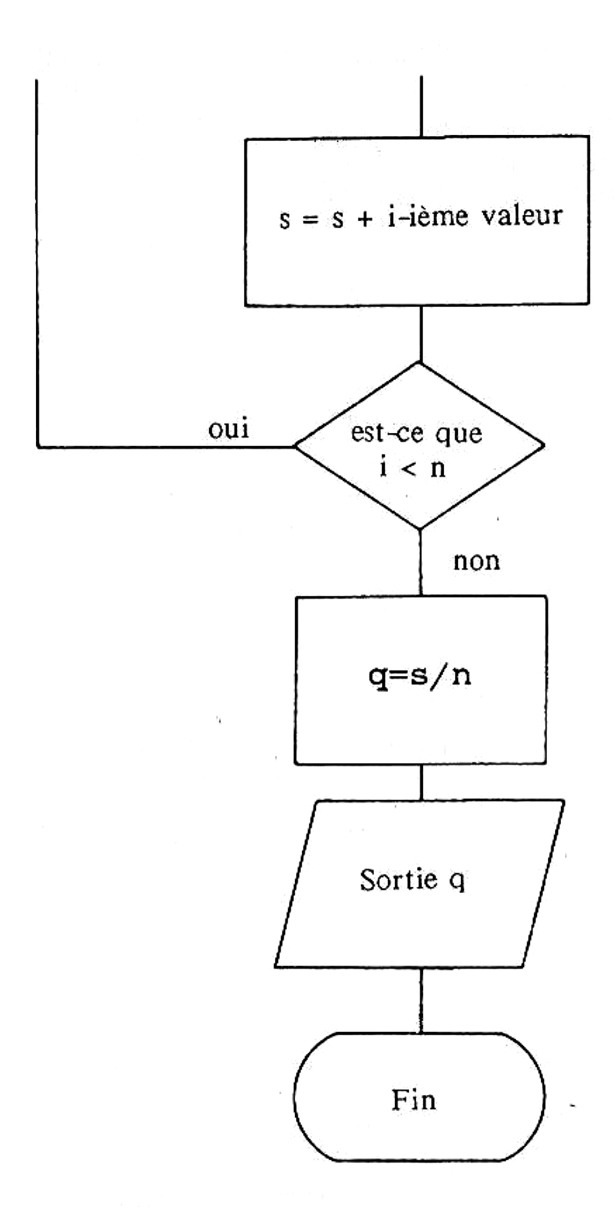
différentes étapes de travail que devra plus tard parcourir le programme en établissant un organigramme. Un organigramme se compose de différents symboles normalisés. La liste suivante vous montre les principaux symboles admis par la norme DIN 66001 :




Limite
Interface
Remarques sur
l'organigramme

Le meilleur moyen d'illustrer l'emploi d'un organigramme dans la programmation est certainement de présenter un exemple. Considérons donc que vous vouliez calculer la moyenne arithmétique w pour n valeurs différentes. L'organigramme pourrait alors se présenter ainsi :




s = s + i-ième valeur
oui
est-ce que
i < n
non
q=s/n
Sortie q
Fin

Il n'est pas difficile de convertir ensuite cet organigramme en instructions BASIC :

```
10 INPUT"Nombre de valeurs ";n
20 DIM w(n)
30 i=i+1
40 PRINT "Valeur";i;:INPUT w(i)
50 s=s+w(i)
60 IF i<n THEN 30
70 q=s/n
80 PRINT"q =";q
90 END
```

❒ **Les variables utilisées ici ont la signification suivante :**

n	Nombre de valeurs
i	Index
w(i)	Les n valeurs différentes
s	Somme des n valeurs
q	La moyenne arithmétique

Cette version du programme colle d'assez près à l'organigramme présenté ci-dessus. Le même programme pourrait cependant être organisé de façon plus efficace :

```
10 INPUT"Nombre de valeurs ";n:DIM w(n)
20 FOR i=1 TO n
30 PRINT "Valeur ";i;:INPUT w(i)
40 s=s+w(i):NEXT i
50 q=s/n:PRINT"q =";q:END
```

Comme vous le voyez, il y a généralement plusieurs possibilités pour convertir un organigramme en programme. C'est l'expérience qui vous permettra de parvenir chaque fois à une version plus proche de l'idéal. Si vous ne comprenez pas la signification des programmes présentés ici, ne vous inquiétez pas. Au chapitre 3, vous trouverez une explication de chacune des instructions employées.

Si vous avez reproduit l'exemple ci-dessus, le mieux serait que vous essayiez immédiatement de résoudre par vous-même un problème du même type. Utilisez pour cela votre langue maternelle. Ce n'est que plus tard que devra intervenir la traduction des termes que vous employez en instructions BASIC. Il est par ailleurs important que vous entriez suffisamment dans les détails. Il ne suffit pas de se contenter d'inscrire dans une case "calcul". Vous devez préciser ce qu'il s'agit de calculer et comment cela doit être fait. Il est également conseillé de travailler avec un crayon à papier pour que vous puissiez plus facilement apporter des corrections. Veillez d'autre part à laisser suffisamment de place entre les différents symboles. Vous pourrez ainsi rajouter certaines choses le cas échéant. Pour le reste, il est très important que vous fassiez vous-même vos propres expériences. Et n'oubliez pas que les organigrammes n'ont pas été développés spécifiquement pour le langage de programmation BASIC. Ils sont indépendants du langage. C'est ainsi par exemple qu'ils servent aussi dans les centres de calcul pour documenter les programmes et pour en améliorer la lisibilité.

Mais, bien sûr, même le meilleur travail de préparation débouche rarement sur un programme fonctionnant parfaitement dès le départ.

Dès que le problème à résoudre devient plus complexe, vous devez vous attendre à passer beaucoup de temps à tester, c'est-à-dire à mettre au point votre programme.

Un conseil qui peut vous éviter beaucoup de problèmes: donnez un nom à votre programme ou à votre version de programme et notez les particularités de chaque version si vous avez différentes versions d'un même programme. Il est également important de supprimer régulièrement les versions de programmes intermédiaires dont vous n'avez plus besoin. Cela vous facilitera considérablement toute utilisation ultérieure d'un programme, surtout si vous ne travaillez plus sur ce programme pendant un certain laps de temps.

Un autre conseil que nous pouvons vous donner pour la phase de tests est de diviser le programme en sections de programme

(modules) de façon à pouvoir tester chaque module séparément.

Il est recommandé d'utiliser à cet effet l'instruction STOP qui vous permet d'interrompre par exemple l'exécution du programme à la fin de chaque module de façon à demander et à contrôler des résultats intermédiaires.

Il peut également être intéressant de faire afficher avec l'instruction PRINT des valeurs de calcul qui resteraient sinon cachées. Il est très facile d'insérer l'instruction PRINT dans les endroits décisifs d'un programme. L'instruction PRINT constitue un très bon moyen de contrôle du déroulement d'un programme.

Après ces quelques mots sur la recherche et l'élimination des erreurs, nous allons arrêter ici ce chapitre d'introduction. Vous trouverez au chapitre 5.5.4 de plus amples informations sur le traitement des erreurs. On vous y montrera en effet comment vous pouvez optimiser votre traitement des erreurs au moyen de certaines instructions spéciales.

2. Digression : La structure du Basic Amstrad

Ce chapitre est particulièrement recommandé pour tous ceux qui s'intéressent aux méthodes statistiques. Par contre, si vous n'avez pas d'atomes crochus avec ce domaine, nous vous conseillons de débuter en BASIC avec le chapitre 3.

Les novices en BASIC seront certainement les premiers étonnés du nombre d'instructions dont dispose le BASIC AMSTRAD. Le chapitre 3 du manuel d'utilisation présente en effet 168 instructions différentes. L'un ou l'autre d'entre vous se demande donc certainement comment cette multiplicité d'instructions pourrait être réduite, c'est-à-dire comment quelques instructions apparentées pourraient être regroupées sous une catégorie de façon à ce que l'on puisse travailler non plus avec 168 instructions mais avec un nombre de catégories notablement réduites.

Vous savez certainement déjà qu'on peut aisément distinguer des instructions de musique ou des instructions graphiques mais cela ne suffit pas. Il s'agit en effet de classer toutes les instructions par catégorie.

Vous avez peut-être déjà remarqué que, dans le chapitre 3 de votre manuel d'utilisation, toutes les instructions apparentées sont indiquées pour chacune des instructions présentées. Si vous vouliez cependant classer, pour ainsi dire, avec du papier et un crayon, les instructions par catégorie en utilisant ce critère cela vous prendrait beaucoup de temps. Par ailleurs les relations de parenté qui sont indiquées ici ne sont pas réciproques, c'est-à-dire que la parenté d'une instruction x avec une instruction y n'implique pas la parenté de l'instruction y avec l'instruction x. C'est ainsi, par exemple, que l'instruction LIST n'a pas d'instructions apparentées d'après le manuel. Par contre, pour l'instruction RENUM, une parenté avec LIST est indiquée. On peut donc tout à fait considérer qu'il existe un rapport de parenté entre ces deux instructions.

Vous vous demandez peut-être où nous voulons en venir. Peut-être en avez vous déjà une idée : dans le présent ouvrage, vous trouverez un programme qui classe toutes les instructions BASIC AMSTRAD par catégories. Vous trouverez le programme ainsi qu'une description de son déroulement avec des exemples de données au chapitre 9.2 de cet ouvrage. Nous allons nous attacher, dans la suite de ce chapitre, à décrire les principes de base régissant ce programme et surtout à présenter et à interpréter les résultats du programme.

Ce programme se trouve dans le chapitre consacré aux applications car il peut être utilisé ou réécrit pour d'autres éléments (en l'occurrence, chaque instruction BASIC constitue un élément). Nous avons d'ailleurs pensé qu'il était préférable de placer ce programme plutôt vers la fin de l'ouvrage car un lecteur encore 'inexpérimenté' pourrait avoir du mal à comprendre ce programme.

Comme nous l'avons déjà indiqué, toutes les instructions BASIC AMSTRAD, telles qu'elles sont présentées dans le chapitre 3 de votre manuel d'utilisation, constituent les données de départ qui seront exploitées par ce programme d'analyse. A cet égard, la signification intrinsèque des différentes instructions ne joue aucun rôle et seuls comptent les rapports de parenté entre les instructions. Le critère choisi pour ces relations de parenté est constitué par les listes présentées à la fin de la description de chaque instruction. Comme il s'agit là d'une opération de classement et de formation de groupes, on pourrait également parler d'une opération d'analyse de cluster. Il ne nous est cependant pas possible de décrire ici les modalités d'une analyse de cluster en tant qu'opération statistique multivariantes.

❐ Le principe de base est le suivant :

Au début de l'opération, chaque instruction représente avec les instructions qui lui sont apparentées une catégorie indépendante. Ces catégories seront par la suite également appelées enregistrements (c'est-à-dire instruction plus instructions apparentées). Chaque fois qu'une instruction se présentera aussi bien dans une catégorie que dans une autre,

ces deux catégories seront réunies et ne formeront plus, pour la suite du déroulement de l'opération, qu'une catégorie unique.

Cette procédure de classement se poursuit jusqu'à ce que chaque catégorie ne contienne plus que des instructions n'apparaissant dans aucune autre catégorie. Le résultat est alors un nombre déterminé de catégories différentes dont chacune contiendra des instructions apparentées ou 'semblables' formant une partie plus ou moins fermée du jeu d'instructions du BASIC. Il faudra ensuite bien sûr interpréter chaque catégorie, c'est-à-dire choisir des titres 'adéquats' pour chacune.

Voilà pour la théorie. Venons-en aux résultats concrets. Notons au passage que le CPC met 17 minutes et 30 secondes pour traiter le programme. Mais si vous étudiez de plus près le chapitre 10.2, vous comprendrez pourquoi.

❐ **Voici maintenant les résultats du programme :**

Catégorie 1

CAT

Catégorie 2

CHR$	ASC

Catégorie 3

CLEAR

Catégorie 4

DEF FN

Catégorie 5

DEFSTR	DEFINT
DEFREAL	

Catégorie 6

ERASE	DIM

Catégorie 7

IF	ELSE
GOTO	THEN

Catégorie 8

INSTR

Catégorie 9

JOY	CLEAR INPUT
INKEY	INKEY$

Catégorie 10

LEN

Catégorie 11

LET

Catégorie 12

LOG 10	EXP
LOG	

Catégorie 13

MIN	MAX

Catégorie 14

MOD

Catégorie 15

NEW

Catégorie 16

NEXT	FOR
STEP	TO

Catégorie 17

ON GOTO

Catégorie 18

OPENIN	CLOSEIN
EOF	

Catégorie 19

POKE	PEEK

Catégorie 20

REM

Catégorie 21

RESTORE	DATA
READ	

Catégorie 22

RESUME NEXT	DERR
ERL	ERR
ERROR	ON ERROR GOTO
RESUME	

Catégorie 23

RIGHT$	MID$
LEFT$	

Catégorie 24

RND	RANDOMIZE

Catégorie 25

SPEED KEY	KEY DEF
KEY	

Catégorie 26

SQR

Catégorie 27

STOP	CONT
END	SPEED WRITE
OPENOUT	SAVE
CLOSEOUT	CHAIN
CHAIN MERGE	LOAD
MERGE	RUN
RENUM	DELETE
LIST	EDIT
AUTO	

Catégorie 28

SYMBOL AFTER	HIMEM
MEMORY	SYMBOL
FRE	

Catégorie 29

TAN	ATN
COS	DEG
RAD	SIN
PI	

Catégorie 30

TRON	TROFF

Catégorie 31

UPPER$	LOWER$

Catégorie 32

VAL	STR$
UNT	CINT
FIX	INT
ROUND	BIN$
DEC$	HEX$
SGN	ABS
PRINT USING	CREAL
CALL	

Catégorie 33

WAIT	INP
OUT	

Catégorie 34

WHILE	TIME
WEND	AFTER
EVERY	SQ
ON SQ GOSUB	SOUND
ENT	ENV
RELEASE	RETURN
GOSUB	REMAIN
DI	EI
ON GOSUB	ON BREAK STOP
ON BREAK CONT	ON BREAK GOSUB

Catégorie 35

WRITE	INPUT
LINE INPUT	

Catégorie 36

XOR	AND
OR	NOT

Catégorie 37

YPOS	MOVE
MOVER	ORIGIN
XPOS	WINDOW SWAP
WINDOW	WIDTH
POS	VPOS
TESTR	TEST
TAGOFF	TAG
SPEED INK	BORDER
INK	PLOTR
GRAPHICS PEN	PLOT
PEN	PAPER
GRAPHICS PAPER	
MODE	MASK
DRAW	DRAWR
LOCATE	CURSOR
COPYCHR$	CLG
CLS	FILL
FRAME	

Catégorie 38

ZONE	PRINT
STRING$	SPACE$
SPC	TAB
USING	

Le nombre total d'instructions présentées ici n'est pas de 168 mais de 179. Cela s'explique par le fait qu'il n'existe pas d'enregistrements particuliers pour quelques-unes des instructions 'apparentées' (TROFF par exemple).

D'ailleurs, ces résultats ne sont nullement 'idéaux' car de nombreuses catégories ne présentent qu'une instruction alors que d'autres en comportent un grand nombre. Cela tient pour une part aux données de départ car, du point du vue de l'analyse, quelques catégories semblent inclassables. Dans certaines catégories comptant de nombreux éléments, a joué le phénomène d'effet de 'chaîne' qu'il nous faudra encore expliquer au chapitre 10.2 car il a son origine dans la logique de la procédure de classement.

Ces résultats peuvent cependant être interprétés et être rangés sous différents termes génériques. Il est toutefois nécessaire pour cela de réunir les différentes catégories. Voici donc une table qui récapitule les catégories et leur affecte des termes génériques. Vous trouverez en outre, dans la troisième colonne, les numéros de chapitre dans lesquels seront traitées ou expliquées l'essentiel des instructions appartenant à chaque catégorie. Comparez un peu cette liste des catégories avec la structure du présent ouvrage. Vous verrez que, même sans disposer de quelconques connaissances en BASIC, vous pourriez ainsi écrire tout au moins la table des matières d'un livre sur le BASIC. Le programme utilisé ne constitue certes pas un exemple d'intelligence artificielle, pour reprendre une notion très prisée, mais il constitue néammoins un aide très appréciable.

Catégorie	Terme générique	Chapitre
1,15,20,27	Organisation du travail sur l'ordinateur et sorties de fichiers	2.2,6.5.5, 9
2	Le code ASCII	5.3,7.4
3,5,11, 13,14,32	Manipulation et définition de nombres et de variables	4.2,5.4, 6.2,6.5.1
4	Définition de fonctions	7.8
6	Réservation de place mémoire	4.2,4.5
7,17	Conditions, sauts de programme conditionnels & inconditionnels	4.3
8,23,10,31	Traitement des chaînes	6.2
9,35	Entrées	4.5,7.10, 9.2.1
12,26,29	Puissances, logarithmes, racines carrées et trigonométrie	4.2,5.4, 7.7,7.8
16	Boucles de programme	4.4
18	Entrées disquettes	9
19	Traitement de places mémoire	5.2,5.7, 6.5.5
21	Préparations de données	4.5
22,30	Traitement des erreurs	6.5.4
24	Nombres aléatoires	6.5.3
25	Définition des touches	6.3.1

28	Organisation de la place mémoire Définition de caractères	5.6, 6.3.2
33	Traitement d'interface	
34	Sous-programmes, interruptions de programme (BREAK et horloge) Musique et répétitions conditionnelles de sections de programme	4.4,4.6, 8
36	Logique	4.3
37	Graphisme, couleur, fenêtre Positionnement du curseur	6.4,7 4.7
38	Les sorties et leurs formats	4.7, 9.2.1

3. Elément de base du Basic

3.1. Note Préliminaire

Ce chapitre aborde des éléments tout à fait essentiels du langage qui sont utilisés dans la plupart des programmes BASIC et qui sont d'une valeur inappréciable. Il est nécessaire de comprendre ces instructions pour comprendre les chapitres suivants.

Mais comme cet ouvrage n'est cependant pas destiné aux tout-débutants, les instructions concernées ne sont expliquées que de manière assez succincte. Ici aussi, le principe est que nous voulons éviter autant que possible de répéter des explications qui se trouvent déjà dans votre manuel d'utilisation. Nous conseillons donc plutôt aux débutants en BASIC d'étudier ce chapitre en utilisant pleinement le manuel d'utilisation. Si vous avez déjà une certaine pratique du BASIC, alors vous pouvez bien sûr passer sans regret sur ce chapitre. Notez toutefois que nous abordons aussi dans ce chapitre certaines 'particularités' du BASIC AMSTRAD.

Encore un conseil pour le débutant 'total' : si les explications données dans ce chapitre ne vous suffisent pas, alors n'oubliez pas que les éléments du langage abordés ici seront utilisés dans de nombreux programmes des chapitres suivants. Comme ces programmes sont généralement commentés de façon très complète, il vous offriront une occasion supplémentaire d'éclaircir le sens de certaines instructions.

3.2. Les Variables et L'Arithmétique

Comme vous aurez encore l'occasion de le constater, il est souvent très utile de pouvoir attribuer des noms à certains emplacements de la mémoire de l'ordinateur. En effet, vous pouvez ensuite stocker sous ces noms, que nous appellerons désormais variables, différents nombres ou même des textes.

❐ On distingue trois types de variables :

❶ Variables réelles

❷ Variables entières

❸ Variables de texte

Alors que les variables réelles peuvent recevoir n'importe quelle valeur à l'intérieur des limites de l'ordinateur (qui sont de +/- 1.7E38, soit 37 zéros après le 17), le point étant utilisé à la place de la virgule décimale, les variables entières n'admettent que des valeurs entre -32768 et +32767 inclus. Les variables de texte reçoivent par contre comme valeurs des suites de symboles (qui peuvent comporter de 0 à 255 caractères) et qui doivent être placées entre guillemets.

Une valeur est attribuée à une variable lorsque vous écrivez par exemple, dans un programme ou même en mode direct : aa=5 ou w3$="qui5" ou encore a=b. L'instruction LET qui était nécessaire dans les dialectes antérieurs du BASIC est totalement superflue en BASIC AMSTRAD. Si vous voulez annuler à nouveau les valeurs de toutes les variables, utilisez simplement l'instruction CLEAR.

Indiquons ici que les instructions DEFINT, DEFREAL et DEFSTR vous permettent de fixer les types de variables pour des listes entières de variables. Les fonctions STR$ et VAL vous permettent par ailleurs de convertir une variable numérique en une chaîne décimale et inversement (voyez également à ce sujet le chapitre 5.2.1).

Les variables présentées jusqu'ici ne représentent chaque fois qu'une valeur. Si vous voulez stocker plusieurs valeurs sous un même nom, vous devez doter la variable d'un index qui vous permettra d'appeler les différentes valeurs de la variable. Cet index figure entre parenthèses à la suite du nom de variable, par exemple x(0), x(1), x(2), x(3) ... x(10). Une telle variable est appelée tableau ou variable indexée. On emploie aussi le terme anglais 'Array'. L'ordinateur considère que la valeur-limite pour l'index est de 10 à moins que vous ne précisiez une autre limite.

Si vous avez besoin de plus de 11 emplacements pour une variable, il est nécessaire de procéder à un dimensionnement, c'est-à-dire à une réservation de place mémoire (voir le chapitre 3.5).

Il nous reste à indiquer que l'index peut être lui-même une variable, soit par exemple x(i). Il est en outre possible d'avoir une double indexation ou une indexation encore plus complexe, soit par exemple x(i,j) ou x(i,j,k). Ici également, un dimensionnement est nécessaire chaque fois que vous avez besoin de plus de 11 places mémoire au total pour chaque index.

La double indexation est très souvent utilisée car elle permet de représenter très aisément des éléments de tableaux ou de matrices. Une variable doublement indexée x(i,j) avec des variables de comptage i de 1 à 3 et j de 1 à 4 pourrait par exemple contenir les 12 éléments suivants :

```
x(1,1),      x(1,2),      x(1,3),      x(1,4)
x(2,1),      x(2,2),      x(2,3),      x(2,4)
x(3,1),      x(3,2),      x(3,3),      x(3,4)
```

On peut aussi attribuer des expressions arithmétiques à des variables. Ces expressions arithmétiques sont bâties à partir de nombres et/ou de variables et d'opérateurs arithmétiques. Lorsqu'on utilise différents opérateurs dans une même expression, on doit tenir compte du rang de priorité de chaque opérateur. On utilise les parenthèses pour changer le rang de priorité.

Voici un récapitulatif de la liste des rangs de priorité des opérateurs mathématiques les plus importants :

( )	Mise entre parenthèses
^	Elévation à la puissance
* et /	Multiplication et division
+ et -	Addition et soustraction

Pour conclure ce chapitre, indiquons encore que des valeurs de fonctions peuvent être également attribuées aux différentes variables.

Nous ne faisons pas allusion ici aux fonctions définies par l'utilisateur (voir DEF FN, chapitre 6.8) mais plutôt aux fonctions mathématiques intégrées telles que SQR, LOG ou SIN. La forme générale d'un appel de fonction est :

```
Nom de variable = nom de fonction (argument)
```

Si vous voulez connaître les arguments autorisés pour chaque fonction, nous vous invitons à vous reporter pour cela à votre manuel d'utilisation. Dans tous les cas, la valeur de fonction est calculée à partir de l'argument puis affectée à la variable placée à gauche du signe égale, par exemple :

```
a=SQR(9), c'est-à-dire a=3.
```

3.3. Conditions et Branchements

Un programme BASIC est traité en suivant l'ordre des numéros de ligne à moins que des branchements, également appelés sauts, n'apparaissent dans le programme. Si l'on excepte la technique des sous-programmes, ces sauts représentent la seule possibilité de s'écarter de l'ordre prescrit par les numéros de ligne.

Il faut cependant distinguer entre les sauts conditionnels et inconditionnels. L'instruction de saut inconditionnel GOTO ne nécessite par exemple pas beaucoup d'explications. Evoquons cependant la possibilité de faire dépendre le saut, avec ON ... GOTO de la valeur d'une variable. Cette possibilité sera notamment utilisée dans ce qu'on appelle les menus. Par contre, le lecteur inexpérimenté risque d'avoir plus de difficulté à bien comprendre ce que représente la condition dans une instruction de saut conditionnel.

Le saut conditionnel est programmé avec l'instruction IF. Sa forme générale est :

```
IF Condition THEN Option (ELSE Option)
```

Si la condition est remplie, l'option placée après THEN sera exécutée. Cette option peut être un numéro de ligne à laquelle il faut sauter ou bien n'importe quelle autre instruction. Si la condition n'est pas remplie, l'exécution du programme passe à la ligne de programme suivante ou bien l'option après ELSE est exécutée s'il y en a une. Si l'instruction IF est encore suivie d'autres instructions, celles-ci ne seront exécutées que si la condition est remplie.

Ce que nous avons jusqu'ici désigné sous le terme général de condition n'est rien d'autre que ce qu'on appelle une expression logique. On peut se représenter une telle expression comme une question à laquelle correspondrait la réponse 'oui'. Si la réponse est 'non', c'est que la condition n'est pas remplie.

Une expression logique telle que celle qui peut être utilisée ici se compose de nombres et/ou de variables et/ou d'expressions arithmétiques ainsi que d'opérateurs de comparaison et d'opérateurs logiques.

Vous diposez à cet égard des opérateurs de comparaison suivants :

=	égal à
<>	différent de
>	supérieur à
<	inférieur à
>=	supérieur ou égal à
<=	inférieur ou égal à

L'expression 2<3 sera par exemple vraie (réponse 'oui'), l'expression 3*4<>12 sera par contre fausse (réponse 'non').

Il convient de dire à ce sujet quelques mots sur les particularités de l'emploi du signe égale dans la terminologie du traitement électronique des données. En effet, en algèbre, l'instruction de calcul x=x+1 serait obligatoirement fausse alors qu'elle est parfaitement admise en BASIC. Il ne s'agit pas en effet, en

BASIC, d'une équation, c'est-à-dire d'une affirmation, mais d'un ordre qui indique que x devra maintenant recevoir la valeur précédente de x plus 1.

Les novices en traitement électronique des données doivent bien sûr retenir cette particularité qui s'applique également avec les autres opérateurs de comparaison.

Les opérateurs logiques sont dérivés de l'algèbre de Boole. Ils sont notamment employés dans la technique digitale électronique.

Votre CPC connaît quatre opérateurs logiques, AND, NOT, OR et XOR. C'est ainsi par exemple que l'affirmation x>10 AND x<12 ne sera vraie que si x est égal à 11. Dans tous les autres cas elle sera fausse.

Le mieux est que vous fassiez quelques essais avec ces opérateurs tant que vous n'en maîtriserez pas pleinement l'emploi. Votre CPC vous dira de toute façon s'ils sont placés de manière 'correcte' ou 'incorrecte'.

Pour terminer, disons encore quelques mots sur les rapports des différents opérateurs dans une affirmation. A cet égard, il ne faut pas tenir compte uniquement des opérateurs présentés dans ce chapitre mais également des opérateurs arithmétiques dont il était question au chapitre précédent.

Tous les opérateurs sont traités d'après un rang de priorité déterminé, et non dans l'ordre dans lequel ils se présentent, de même que les opérateurs arithmétiques entre eux. Les opérateurs arithmétiques ont toujours le plus haut rang de priorité, suivis des opérateurs de comparaison puis des opérateurs logiques qui ont donc le rang de priorité le plus bas.

3.4. Boucles de programmation

Il est souvent nécessaire dans un programme de répéter certaines parties du programme. Cela peut parfaitement être programmé avec les instructions de saut dont nous avons déjà

parlé. Si vous vouliez par exemple calculer la somme de tous les nombres entiers de 1 à 100, le programme approprié pourrait se présenter ainsi :

```
10 i=i+1
20 s=s+i
30 IF i<100 THEN 10
40 PRINT s
50 END
```

La variable i recevra ici les différentes valeurs entières de 1 à 100 et chaque valeur provisoire de i sera ajoutée dans la variable s à la somme déjà formée par s. La partie du programme formée par les lignes 10-30 sera donc parcourue autant de fois que la variable i sera supérieure ou égale à 1 et inférieure ou égale à 100. Le résultat pourra alors être sorti en ligne 40 et le programme se terminera en ligne 50.

Chaque fois qu'une opération déterminée doit être exécutée plusieurs fois, on parle d'une boucle. Les instructions FOR et NEXT permettent de construire des boucles plus facilement. L'exemple ci-dessus se présenterait alors différemment :

```
10 FOR i=1 TO 100
20 s=s+i
30 NEXT i
40 PRINT s
50 END
```

Lorsque l'exécution du programme parvient à l'instruction FOR, la variable de comptage i est fixée sur la valeur 1 et la ligne 20 est alors exécutée avec cette valeur de i. Lorsque l'instruction NEXT est atteinte, i est augmentée du pas de progression indiqué par STEP, 1 en l'occurrence, car nous n'avons rien indiqué avec STEP. La ligne 20 est alors à nouveau traitée avec la nouvelle valeur de i. Cette opération se répète jusqu'à ce que la valeur-limite 100 fixée à la ligne 10 ait été atteinte lors de l'exécution de l'instruction NEXT. Ce n'est qu'alors que les lignes 40 et 50 seront traitées.

Indiquons encore qu'il est possible de sauter à l'intérieur d'une boucle et qu'il est également possible de quitter des boucles (mais attention à la valeur de la variable de comptage !).

Par ailleurs, le pas de progression indiqué optionnellement avec STEP peut aussi être négatif.

Il est également possible d'imbriquer des boucles. Une telle imbrication sera par exemple nécessaire pour affecter des valeurs à une variable doublement indexée. Reprenons l'exemple du chapitre précédent où la variable x(i,j) avait été présentée avec i variant de 1 à 3 et j de 1 à 4. Si vous vouliez affecter des valeurs à cette variable au moyen de l'instruction INPUT (voir le chapitre suivant), vous pourriez utiliser les instructions de programme suivantes :

```
FOR i=1 TO 3
FOR j=1 TO 4
INPUT x(i,j)
NEXT j
NEXT i
```

Notez qu'il n'est pas nécessaire d'indiquer le nom de la variable dans une instruction NEXT (le BASIC trouvera automatiquement à quel FOR se rapporte un NEXT donné) mais cela facilitera la compréhension du programme si vous le faites quand même, surtout pour les boucles imbriquées.

Les instructions WHILE et WEND vous offrent une possibilité totalement différente de construire des boucles. La section de programme placée entre WHILE (début de la boucle) et WEND (fin de la boucle) sera ici exécutée tant que l'expression logique placée dans l'instruction WHILE sera remplie. WHILE et WEND permet en quelque sorte de réunir les fonctions des instructions IF ... THEN et FOR ... NEXT. Lorsque le nombre de parcours de la boucle ne peut être fixé d'avance, il est préférable de travailler avec WHILE et WEND. Cela permet de programmer très facilement des opérations dépendant de certains délais (voir le chapitre 3.6). Notre problème de calcul de la somme peut d'ailleurs parfaitement être résolu avec WHILE et WEND :

```
10 WHILE i<100
20 i=i+1
30 s=s+i
40 WEND
50 PRINT s
60 END
```

3.5. Entrée de données et réservation de mémoire

Nous n'avons pas grand chose à ajouter en ce qui concerne la réservation de place en mémoire (voir chapitre 3.2). Chaque fois que vous avez besoin pour un tableau ou une matrice de plus des 11 emplacements fixés automatiquement, vous devez réserver de la place en mémoire au moyen de l'instruction DIM. Notez que l'instruction DIM doit être placée autant que possible avant toute opération de calcul. Si un tableau devient inutile dans le déroulement du programme, vous pouvez le supprimer avec ERASE. La place mémoire qu'il occupait est alors à nouveau disponible.

Venons-en au point important que constitue l'entrée des données. Lorsque vous voulez mettre à la disposition de l'ordinateur des masses importantes de données, il est recommandé de le faire avec les instructions DATA et READ. Les valeurs préparées avec l'instruction DATA seront chargées dans l'ordinateur avec l'instruction READ ou affectées aux variables de l'instruction READ (Les données sont lues de gauche à droite). Les valeurs de données et les variables sont chaque fois séparées entre elles par des virgules. Notez que vous ne devez pas indiquer dans l'instruction READ plus de variables qu'il n'y a de valeurs dans l'instruction DATA. Par ailleurs, si vous mélangez les types de variables, vous devez veiller à ce que la variable utilisée pour lire une donnée soit bien du même type que cette donnée. Si une chaîne de caractères doit contenir comme valeur de données des virgules ou des doubles points, cette chaîne doit être placée entre guillemets car les virgules servent sinon à séparer les différentes valeurs de données.

L'ordinateur lit normalement les données à la suite de DATA dans l'ordre dans lequel elles se présentent. Mais si, après que certaines données aient déjà été lues, vous voulez que le programme lise à nouveau les premières données et non les données suivantes, vous devez restaurer la masse de données. Cela est obtenu avec l'instruction RESTORE.

Cette instruction vous permet même d'accéder à des lignes de programme DATA déterminées (voir le manuel d'utilisation). Vous trouverez au chapitre 7.5 un exemple de cet emploi assez rare de l'instruction RESTORE. La masse de données est en effet continuellement restaurée dans le programme 'Composition' de façon à ce que la sortie de la mélodie programmée soit répétée sans interruption.

L'instruction INPUT vous offre une autre possibilité d'entrée des données. Lorsque le programme atteint cette instruction, le traitement du programme s'interrompt et l'utilisateur peut entrer des données à affecter aux variables placées à la suite de INPUT (s'il y en a plusieurs, les variables dans le programme et les données entrées devront être séparées entre elles par des virgules). Ici également il faut bien entendu tenir compte du type de variable. A la différence de READ et DATA, les données sont ici entrées dans un dialogue entre ordinateur et utilisateur. Cependant cette méthode d'entrée de données est peu adaptée pour des masses importantes de données car des fautes de frappe peuvent aisément se produire. Si vous ne savez pas alors à quelle variable vous avez affecté une valeur incorrecte (de façon à corriger cette valeur en mode direct après avoir interrompu le programme), il vous faut répéter tout le travail d'entrée ce qui peut parfois occasionner une perte de temps considérable.

Si une variable de texte doit contenir des virgules, le texte doit être placé entre guillemets lorsqu'il est entré par l'utilisateur. Ce problème peut d'ailleurs être résolu de façon plus simple si vous employez dans le programme l'instruction LINE INPUT. Dans ce cas, vous pouvez même placer des guillemets dans une variable de texte. Pour plus de détails sur la comparaison entre INPUT et LINE INPUT, notamment en ce qui concerne l'entrée

de données à partir de la disquette, nous vous invitons à vous reporter au chapitre 8.2.1.

Contrairement à l'instruction INPUT, INKEY$ n'arrête pas le programme. Cette instruction ne permet d'entrer qu'un caractère à la fois, lettre, chiffre ou autre. L'ordinateur interroge en effet le clavier pour voir si une touche de texte a été enfoncée. Nous allons illustrer cette instruction par un exemple. Supposons que vous ayez à sortir des résultats dépassant la taille de l'écran. Il vous faudra alors construire une boucle d'attente qui sera atteinte chaque fois que l'écran sera plein. Cela est souvent programmé avec la construction 'attendre touche' et cela se présente ainsi :

```
500 PRINT"Veuillez frapper une touche"
510 x$=INKEY$:IF x$="" THEN 510
```

Lorsque l'ordinateur atteindra cette partie du programme, le programme sera arrêté aussi longtemps qu'aucune touche ne sera enfoncée car l'ordinateur sort une chaîne vide "" si aucune touche n'est enfoncée. Vous pouvez bien sûr modifier à votre guise l'instruction IF (en fonction des combinaisons de touches que vous attendez). Il sera également souvent intéressant de combiner plusieurs interrogations IF.

Si vous voulez demander directement au clavier quelle touche est enfoncée, vous pouvez utiliser l'instruction INKEY. L'argument de cette instruction est toujours un numéro de touche (voyez sur votre ordinateur la disposition placée à droite du clavier). Si c'est bien la touche correspondante qui est enfoncée, l'ordinateur vous renvoie le nombre 0 et 1 si elle n'est pas enfoncée. La combinaison avec les touches SHIFT et CONTROL est également possible (voir le manuel d'utilisation).

L'exemple ci-dessus peut donc être modifié ainsi :

```
500 PRINT"Veuillez appuyer sur la touche espace"
510 IF INKEY(47)=-1 THEN 510
```

Si dans cet exemple vous actionnez quelques autres touches avant la touche espace, ces caractères qui figureront encore

dans le buffer (c'est-à-dire la mémoire provisoire) du clavier apparaîtront ensuite sur l'écran. Le buffer clavier est une petite mémoire dans laquelle peuvent être stockés provisoirement jusqu'à 20 caractères. CLEAR INPUT vous permet de le vider. Tapez simplement :

```
520 CLEAR INPUT
```

et lancez à nouveau le programme. Vous verrez que cette instruction annule les informations contenues dans le buffer.

Indiquons simplement en conclusion de ce chapitre que vous pouvez également utiliser un joystick (manche à balai) connecté à votre ordinateur pour entrer des informations. Le chapitre 6.10 vous en dira plus sur cette possibilité.

3.6. Les sous-programmes et les interruptions

Pour résoudre certains problèmes, il est parfois nécessaire de faire exécuter plusieurs fois certaines tâches partielles. Pour vous éviter de devoir programmer plusieurs fois ces tâches partielles, vous avez la possibilité de développer un programme séparé, appelé sous-programme, que vous pourrez appeler ensuite plusieurs fois à partir du programme proprement dit appelé programme principal. Ces sous-programmes peuvent figurer en n'importe quel endroit du programme principal et ils sont appelés avec l'instruction GOSUB. Un sous-programme doit se terminer par l'instruction RETURN. On sautera alors à nouveau au programme principal et plus précisément à l'instruction suivant l'instruction GOSUB.

Pour mieux vous faire comprendre ce que nous venons d'expliquer, nous allons vous présenter un exemple. Supposons donc que vous ayez une petite entreprise et que vous vouliez calculer pour plusieurs valeurs la moyenne arithmétique pour une période comptable donnée. Il conviendra bien sûr de ne programmer qu'une fois le calcul de la moyenne et ce dans un sous-programme que vous pourrez ensuite appeler plusieurs fois.

La solution de ce problème pourrait se présenter ainsi :

```
10 DATA 10.5,12.95,19.4:REM Prix
20 DATA 145.25,90.23,311:REM Chiffres d'affaires
30 FOR i=1 TO 3
40 READ p(i)
50 NEXT
60 n=3:FOR i=1 TO 3:x(i)=p(i):NEXT:GOSUB 1000
70 PRINT"Prix moyen  =";d
80 FOR i=1 TO 3
90 READ u(i)
100 NEXT
110 n=3:FOR i=1 TO 3:x(i)=u(i):NEXT:GOSUB 1000
120 PRINT"Chiffre d'affaires =";d
130 END
1000 REM Sous-programme
1010 s=0
1020 FOR i=1 TO n
1030 s=s+x(i)
1040 NEXT
1050 d=s/n
1060 RETURN
```

Les lignes 10 et 20 préparent trois prix et trois chiffres d'affaires comme données de départ. Ces données seront chargées chaque fois dans les lignes 30 à 50 et 80 à 100 (p(i) = prix, u(i) = chiffres d'affaires). Comme le sous-programme (lignes 1000 - 1060) calcule la moyenne arithmétique seulement pour la variable hypothétique x(i), la variable x(i) devra être fixée égale aux variables p(i) ou u(i) qui contiennent des données, dans les lignes 60 à 110, avant l'appel du sous-programme. En même temps, le sous-programme nécessite une information sur le nombre des données (n=3). Après chaque traitement du sous-programme, les résultats peuvent être alors sortis dans les lignes 70 et 120. Comme vous le voyez, les résultats sont placés dans la variable d indépendamment des données de départ.

Ce programme peut être étendu pour admettre un nombre quelconque de grandeurs. Pour des applications sérieuses, il conviendrait toutefois d'ajouter quelques étapes de calcul

supplémentaires comme par exemple le calcul des chiffres d'affaires à partir des grandeurs 'prix' et 'quantités'.

Les sous-programmes peuvent aussi être imbriqués (appel d'un sous-programme par un autre sous-programme).

La technique des sous-programmes permet de construire de très belles structures de programmation. C'est ainsi que vous pouvez placer au début du programme la commande du déroulement du programme et programmer l'ensemble du corps véritable du programme sous forme de sous-programmes. Dans ce cas, il peut fort bien arriver que les sous-programmes appelés avec GOSUB vous fournissent une structure très utile même pour des parties de programme qui ne seront exécutées qu'une seule fois dans le cours du déroulement du programme.

Une structure de programme de ce type est beaucoup plus facile à lire mais elle présente aussi le grand avantage de vous permettre de tester les différents sous-programmes indépendamment les uns des autres. Vous comprenez certainement que cette méthode peut vous permettre de parvenir assez rapidement à la réalisation de programmes en état de fonctionner.

En appuyant deux fois sur la touche ESC, vous pouvez arrêter l'exécution d'un programme. Si vous voulez que cela ne soit pas possible, vous disposez des instructions ON BREAK CONT, ON BREAK GOSUB et ON BREAK STOP (voyez le manuel d'utilisation). Par ailleurs, ON GOSUB vous permet de faire dépendre un appel de sous-programme d'un sélecteur (voyez aussi à ce sujet le chapitre 3.3, ON GOTO).

Une autre possibilité de programmer des interruptions vous est offerte par l'horloge intégrée. Votre CPC dispose en effet d'une horloge en temps réel qui vous permet de commander ou d'interrompre vos programmes en fonction du temps écoulé. A cet effet le BASIC AMSTRAD est en mesure d'exécuter simultanément différentes opérations (multi-tasking). C'est ainsi par exemple que vous pouvez, avec les instructions AFTER et EVERY, appeler des sous-programmes en fonction du

paramètre de durée indiqué. Le déroulement du programme normal se poursuivra donc jusqu'à ce que soit écoulé le délai indiqué. Le petit programme suivant vous montre comment vous pouvez faire appeler un sous-programme toutes les minutes :

```
10 EVERY 3000,0 GOSUB 1000
20 REM Simulation du programme principal
30 FOR i=1 TO 1E+20:NEXT
40 END
1000 REM Sous-programme
1010 j=j+1
1020 PRINT j;"minutes se sont ecoulees
1030 RETURN
```

Le paramètre 3000 de l'instruction EVERY indique ici l'intervalle de temps (en unités de cinquantièmes de seconde, 3000*0.02 = 1 minute). Le paramètre 0 détermine l'horloge utilisée (vous disposez au total de 4 horloges dotées de rangs de priorité différents (voyez le manuel d'utilisation).

Si nécessaire pour le bon déroulement de votre programme, vous pouvez aussi interdire ou autoriser à nouveau les interruptions provoquées par les horloges, grâce aux instructions DI et EI. La fonction REMAIN vous offre en outre la possibilité non seulement de déconnecter une horloge mais encore de vous faire fournir le temps restant.

La fonction TIME vous fournit une autre possibilité d'utiliser l'horloge en temps réel de votre CPC. TIME indique en unités de trois-centièmes de seconde le temps écoulé depuis l'allumage de votre ordinateur. L'emploi de cette fonction peut être notamment très intéressant si vous définissez les valeurs de différentes variables avec la fonction TIME. C'est ainsi par exemple que le programme suivant vous montre comment vous pouvez utiliser votre CPC comme réveil :

```
10 REM Reveil
20 CLS
30 PRINT"Dans combien de temps"
40 PRINT"la sonnerie doit-elle retentir":PRINT
```

```
50 INPUT"Heure   ";st
60 INPUT"Minute  ";mi
70 INPUT"Seconde ";se
80 z=st*3600+mi*60+se+TIME/300
90 IF TIME/300<z THEN 90
100 SOUND 1,200,600,15:GOTO 100
```

Dans ce programme, le délai entré en heures, minutes et secondes est converti en secondes de même que le temps indiqué par la fonction TIME. L'unité de la variable de délai 'z' est donc une seconde. La comparaison entre le temps écoulé et le temps indiqué s'effectue à la ligne 90. Vous pouvez bien sûr organiser autrement cette comparaison. Vous pouvez par ailleurs programmer ce programme de réveil également avec l'instruction AFTER. Mais dans ce cas vous occuperez une des quatre horloges ce qui n'est pas le cas dans le programme ci-dessus. Peut-être pourriez-vous, à titre d'exercice, réécrire ce programme en employant AFTER.

La ligne 100 produit un son jusqu'à ce que la touche ESC ait été appuyée par deux fois. Si vous ne connaissez pas les paramètres de l'instruction SOUND, cela n'est pas très grave. Nous étudierons la programmation de la musique de plus près au chapitre 7.

3.7. Sorties et Formatages

L'instruction PRINT est l'une des instructions BASIC les plus importantes. Dans les exemples de programmes précédents, nous l'avons déjà rencontrée. Elle permet de sortir des résultats ou des commentaires mais elle peut aussi être utilisée pour le calcul (par exemple PRINT 3+5).

Voici maintenant à nouveau récapitulées les règles principales pour l'utilisation de l'instruction PRINT :

Les valeurs à sortir doivent être séparées par des virgules dans l'instruction PRINT si la sortie de chaque valeur doit s'effectuer chaque fois dans la prochaine zone écran. La largeur des zones

écran peut être modifiée avec l'instruction ZONE (la largeur standard est de 13 caractères).

Si vous utilisez le point-virgule comme séparation, la sortie de la prochaine valeur se fera immédiatement à la suite de la sortie précédente.

Une virgule ou un point-virgule à la fin d'une instruction PRINT interdit le passage à la ligne qui est normalement effectué automatiquement après chaque sortie.

Suivant le mode écran sélectionné (avec l'instruction MODE), vous disposez de 20, 40 ou 80 colonnes sur 25 lignes pour vos sorties sur écran. L'instruction LOCATE vous permet d'amener le curseur de texte (nous découvrirons le curseur graphique au chapitre 6) dans un emplacement quelconque de l'écran.

LOCATE a deux paramètres. Le premier indique la colonne voulue et le second la ligne voulue. En liaison avec LOCATE, vous pouvez également copier un caractère avec COPYCHR$ (voyez le manuel d'utilisation).

Outre la possibilité, d'emploi relativement simple, d'insérer des espaces dans une sortie avec PRINT TAB ou PRINT SPC ou encore de modifier des sorties de texte avec SPACE$ ou STRING$ (voyez aussi à ce sujet le chapitre 5.2.1), l'emploi de l'instruction PRINT USING peut vous rendre de grands services lorsque vous voulez donner des indications précises sur le format d'une sortie. Le format est alors déterminé par un modèle de format constitué de ce qu'on appelle des caractères de format. L'instruction PRINT USING est très pratique notamment lorsqu'il s'agit de créer des tableaux où des nombres doivent figurer exactement les uns sous les autres et où des textes ne doivent pas dépasser une longueur donnée.

Pour les différents caractères de format, il convient de distinguer entre les expressions numériques et les textes. Cela est expliqué de manière détaillée dans votre manuel d'utilisation. Tel ou tel lecteur pourrait malgré tout avoir quelques difficultés avec cette instruction. C'est pourquoi le débutant trouvera ici un programme qui lui facilitera le travail

et les essais avec PRINT USING. Après que le programme ait été lancé, l'utilisateur doit décider s'il veut formater un nombre ou un texte. Ce choix s'opère sous la forme d'un petit menu de sélection que vous retrouverez souvent dans cet ouvrage sous une forme semblable. Il faut ensuite entrer un nombre ou un texte ainsi qu'une indication sur le format. La sortie formatée est alors immédiatement exécutée et le menu apparaît à nouveau dès que vous appuyez sur une touche.

Nous vous indiquerons au chapitre 5.5.4 comment vous pouvez améliorer ce programme en interceptant les entrées erronées (ON ERROR GOTO).

```
10 REM Aide au formatage
20 CLS
30 PRINT TAB(22)"Entree":PRINT:PRINT
40 PRINT"Nombre";TAB(25)"1":PRINT
50 PRINT"Texte";TAB(25)"2":PRINT
60 PRINT"Fin du programme";TAB(25)"3":PRINT
70 PRINT:PRINT"Votre choix ?"
80 PRINT:PRINT:PRINT
90 a$=INKEY$
100 IF a$="1" THEN 140
110 IF a$="2" THEN 190
120 IF a$="3" THEN END
130 GOTO 90
140 INPUT"Nombre ";z
150 LINE INPUT"Format ";f$
160 PRINT:PRINT"Sortie formatée :"
170 PRINT:PRINT USING f$;z
180 GOTO 230
190 INPUT"Texte ";s$
200 LINE INPUT"Format ";f$
210 PRINT:PRINT"Sortie formatée :"
220 PRINT:PRINT USING f$;s$
230 LOCATE 7,25
240 PRINT"Veuillez frapper une touche !"
250 a$=INKEY$:IF a$="" THEN 250
260 GOTO 20
```

Liste de variables :

a$	Chaîne indiquant la valeur de la touche enfoncée
f$	Variable de texte pour représenter un modèle de format
s$	Texte à formater
z	Nombre à formater

Description du programme :

10	Commentaire qui ne sera pas sorti sur l'écran.
20	Vidage de l'écran.
30-80	Sortie du menu de sélection.
90-130	Evaluation de la touche enfoncée et saut de programme correspondant (ou fin du programme). En ligne 130 est effectué un retour à la ligne 90 pour assurer une interrogation 'permanente' du clavier, jusqu'à ce qu'une des touches indiquées soit appuyée. Cette section du programme pourrait bien sûr également être écrite de façon plus simple si on utilisait l'instruction INPUT au lieu de l'instruction INKEY$. Dans ce cas toutefois, l'utilisateur serait obligé d'appuyer aussi sur la touche RETURN ou ENTER après avoir entré un nombre et cela ralentirait le déroulement du programme. Une amélioration est cependant encore possible si on essaie d'activer avec l'instruction CURSOR le curseur qui n'est pas visible normalement avec l'instruction INKEY$ (voyez le manuel d'utilisation).
140-170	Après l'entrée d'un nombre et d'un modèle de format, la sortie formatée est effectuée. Le modèle de format est représenté ici par la variable de texte f$. On travaille ici avec LINE INPUT car des virgules peuvent aussi être des caractères de formatage.

180	Dans cet emplacement du programme devraient normalement figurer des instructions pour créer une boucle d'attente et pour revenir à la sélection dans le menu. Comme cependant cela est également nécessaire après la sortie de texte formatée, on se contente ici de sauter aux lignes correspondantes (lignes 230-260). Le programme peut bien sûr également être modifié de façon à ce que les lignes 230 à 260 constituent un sous-programme appelé avec GOSUB.
190-220	Correspondant à la partie du programme chargée des sorties numériques (lignes 140-170) est effectué ici le formatage d'un texte. Si le texte à entrer doit comprendre également des virgules, il vous faut également utiliser l'instruction LINE INPUT pour s$.
230-260	Attente d'une touche (sortie de commentaire dans la ligne inférieure de l'écran) et retour au menu.

4. Hardware et Software - Le Basic et le CPC

4.1. Introduction

Après vous avoir présenté dans le dernier chapitre les éléments de base du BASIC, nous allons maintenant essayer de vous donner une idée du traitement interne d'un programme BASIC. Bien entendu, vous pouvez parfaitement écrire des programmes puissants sans avoir la moindre idée du fonctionnement interne de l'ordinateur mais vos rapports avec votre CPC 6128 se modifieront certainement si vous savez un peu comment il manie ce que vous entrez. D'autre part, vous comprendrez mieux certaines instructions BASIC qui travaillent avec des paramètres dont le sens est fourni par les bits si vous apprenez certaines notions de base sur les nombres binaires, les bits et les octets.

Disons cependant très nettement d'emblée qu'un ouvrage consacré au BASIC ne peut en même temps contenir des explications complètes sur la machine. Il s'agit simplement dans ce chapitre de vous donner des informations pouvant être utiles pour le programmeur BASIC. Si vous vous intéressez toutefois sérieusement à l'apprentissage du langage machine, nous vous invitons à vous reporter à la littérature consacrée à ce thème (par exemple "Le langage machine sur le CPC" de DATA BECKER & MICRO APPLICATION).

4.2. Notions de base du traitement des données

Les différentes instructions BASIC ne sont pas comprises directement par l'ordinateur. Elles doivent d'abord être traduites sous une forme compréhensible pour l'ordinateur. Cette tâche est assumée par ce qu'on appelle l'interpréteur BASIC. Lorsque vous entrez une instruction au clavier et que vous actionnez la touche ENTER ou RETURN, l'instruction est convertie par l'ordinateur en ce qu'on appelle le langage machine. Elle est transmise sous cette forme à l'unité centrale et

ce n'est qu'alors qu'elle est exécutée. L'unité centrale ou Central Processing Unit est en quelque sorte le cerveau de l'ordinateur.

Sur le CPC, il s'agit d'un composant Z 80 qui constitue l'un des processeurs 8 bits les plus répandus dans le domaine de la micro-informatique.

Le véritable traitement des données est donc effectué dans l'unité centrale. Les travaux d'organisation tels que la gestion de la mémoire sont effectués par le système d'exploitation. Mais en tant que programmeur BASIC vous n'avez pas à vous préoccuper de l'adressage des différents emplacements de la mémoire.

Cela ne devient nécessaire qu'à partir du moment où vous voulez utiliser directement le langage machine. Vous devez alors écrire les différents codes machine dans des emplacements mémoire déterminés. Cela peut être fait avec l'instruction POKE. Comme l'ordinateur, comme vous le savez peut-être déjà, ne comprend que les chiffres 0 et 1 (c'est-à-dire les nombres binaires), ces codes ne sont également constitués que de nombres binaires.

L'emploi de ce qu'on appelle un moniteur de langage machine permet toutefois de programmer en langage machine de façon plus rapide et plus pratique. Ce moniteur vous évite en effet d'avoir à POKEr des quantités de codes.

Pour connaître le contenu d'une case mémoire, vous devez employer l'instruction PEEK. PEEK (i) vous restitue la valeur de la case mémoire i. Lorsque vous voulez lancer un programme machine, vous devez utiliser l'instruction CALL. CALL x vous permet ainsi d'appeler à partir du BASIC un programme machine commençant à partir de la case mémoire x.

L'assembleur est un langage de programmation qui vous permet de travailler en employant des abréviations (appelées mnémoniques) à la place des codes d'instructions du langage machine. Les différentes instructions sont alors converties en nombres binaires pour que la machine puisse les comprendre. Notez d'ailleurs que le Z 80 dispose d'un jeu de plus de 600

instructions. Il ne serait bien entendu pas possible d'expliquer dans le cadre de cet ouvrage la signification de chacune de ces instructions.

Nous avons d'ailleurs tout lieu de supposer que les programmeurs BASIC non encore 'confirmés' auront plutôt été plongés dans la perplexité par les courtes explications fournies dans ce chapitre. Mais ne vous laissez pas troubler. Peut-être les chapitres suivants lèveront-ils déjà un certain nombre de difficultés de compréhension.

4.3. Le code ASCII

Comme nous l'avons déjà indiqué, un ordinateur ne peut traiter que des nombres. C'est pourquoi les caractères que vous entrez au clavier sont traduits en un code numérique. Cela se fait en général d'après le code ASCII ou American Standard Code for Information Interchange. Votre manuel d'utilisation vous fournit une liste des différents codes et de leurs significations respectives mais vous pouvez aussi utiliser la fonction ASC pour connaître la valeur ASCII d'un caractère quelconque. Si vous entrez par exemple :

```
PRINT ASC("a")
```

en mode direct, la valeur 97 apparaîtra à l'écran.

Comme la lettre "a" a toujours la valeur 97 en code ASCII, tout récepteur d'informations (imprimante, un autre ordinateur) travaillant aussi avec le code ASCII (pratiquement tous) interprètera le nombre 97 comme la lettre "a". S'il n'y avait pas cette standardisation, l'échange de données entre différents récepteurs de données ne serait pas possible. Ce n'est que grâce à ce standard qu'il est par exemple possible de transmettre des données à travers un modem.

Avec votre CPC, vous pouvez également sauvegarder un programme sous forme d'un fichier ASCII :

```
SAVE "Nom",A1
```

Cette possibilité est surtout intéressante lorsqu'il s'agit de charger un programme dans un traitement de texte.

Les valeurs ASCII 32 à 127 sont utilisées pour des lettres, chiffres et quelques autres caractères. Les valeurs 0 à 31 correspondent à des caractères de commande déterminés dont nous parlerons un peu plus loin. La zone de valeurs de 0 à 127 représente le code ASCII véritablement standard. Les valeurs 128 à 255 présentées dans votre manuel d'utilisation correspondent par contre à des caractères graphiques déterminés de votre CPC 6128.

Pour le programmeur BASIC, l'inverse de l'instruction ASCII constitué par la fonction CHR$ est très intéressant. Si vous entrez par exemple :

```
? CHR$(97)
```

vous verrez apparaître sur l'écran le caractère correspondant, "a".

Le petit programme ci-dessous vous permet de faire sortir le jeu de caractères complet de votre CPC :

```
10 FOR i=32 TO 255
20 PRINT CHR$(i);
30 NEXT
```

Lorsque vous utilisez des caractères graphiques dans des programmes, l'emploi de cette fonction est vivement recommandé car si vous placez dans votre programme des caractères graphiques obtenus avec la touche CONTROL vous risquez d'obtenir un résultat ambigu lorsque vous imprimerez le listing de votre programme (ce qui d'ailleurs est largement fonction de l'imprimante utilisée). Par ailleurs, il n'est pas toujours très facile de trouver un caractère déterminé avec la touche CONTROL.

Si vous entrez par exemple en mode direct l'instruction PRINT puis ensuite un CONTROL 'G' placé entre guillemets, un bip retentira après que vous avez appuyé sur la touche RETURN.

C'est que le caractère spécial placé entre guillemets déclenche une fonction spéciale que vous pouvez tout aussi bien appeler avec :

```
PRINT CHR$(7)
```

Cette fonction fait partie des caractères de commande évoqués plus haut. Votre manuel vous fournit une liste détaillée de ces codes de commande. Si la touche HOME qu'on trouve sur d'autres ordinateurs vous manque, alors entrez simplement :

```
PRINT CHR$(30)
```

Vous verrez que le curseur sera alors placé dans l'angle supérieur gauche de l'écran.

La fonction CHR$ peut être très utile notamment lorsqu'il s'agit d'appeler dans le cadre d'un programme des fonctions de l'ordinateur qui sont normalement déclenchées à travers le clavier. Songez par exemple à la commande du curseur ou au Carriage Return (retour de chariot) que vous pouvez appeler également avec :

```
PRINT CHR$(13)
```

4.4. Les Systèmes Numériques

L'unité centrale de votre CPC ne comprend que les nombres binaires qui sont des nombres composés uniquement des chiffres 0 et 1. Un ordinateur se compose en effet d'un grand nombre de commutateurs qui ne connaissent que deux états 'courant passe' (chiffre 1) et 'courant ne passe pas' (chiffre 0).

Avant que nous ne nous intéressions de plus près au système binaire, essayons de définir clairement comment est organisé le système décimal que nous connaissons si bien. Examinons un exemple:

Le nombre 2375 peut aussi être représenté de la manière suivante :

```
2375 = 2*1000 + 3*100 + 7*10 + 5*1
```

avec :

```
2*1000 = 2*10^3
3*100  = 3*10^2
7*10   = 7*10^1
5*1    = 5*10^0
```

A chaque chiffre du nombre décimal 2375 est donc attribuée une puissance de dix déterminée. C'est de cette façon que nous pouvons représenter tous les nombres décimaux imaginables.

En système binaire, on procède exactement de la même façon si ce n'est qu'on ne travaille plus alors avec la base 10 mais avec la base 2. Voyons un autre exemple :

Le nombre binaire 10101010 sera égal au nombre décimal :

```
1*2^7+0*2^6+1*2^5+0*2^4+1*2^3+0*2^2+1*2^1+0*2^0
```

Les différents chiffres du nombre binaire 10101010 correspondent à une puissance de deux déterminée de même que les chiffres d'un nombre décimal à une puissance de dix déterminée.

Si vous additionnez les différents produits partiels, vous obtenez pour le nombre binaire 10101010 le nombre décimal 170. Vous pouvez toutefois faire exécuter cette conversion de façon beaucoup plus simple, en entrant, en mode direct :

```
PRINT &x10101010
```

Après que vous appuyé sur la touche RETURN, vous verrez apparaître sur l'écran le nombre décimal 170.

Si vous voulez convertir un nombre décimal en un nombre binaire, vous pouvez utiliser tout simplement la fonction BIN$. Il serait toutefois bon que vous essayiez auparavant de comprendre une bonne fois comment fonctionne cette conversion : vous devez d'abord rechercher la puissance de 2 la

plus élevée faisant partie du nombre décimal à convertir. Comme nous l'avons vu plus haut, pour le nombre décimal 170, ce serait 2^7. Cette valeur (128) doit alors être ôtée du nombre décimal à convertir. Vous répétez ensuite la même opération avec le reste (42). Si la puissance de 2 immédiatement inférieure (2^6 en l'occurrence) n'est pas contenue dans le reste (comme c'est d'ailleurs le cas avec 42), vous placez bien sûr un 0 en regard de la puissance correspondante. Au chapitre 7.3, vous trouverez un programme qui effectue des conversions de cette manière.

Cela vous montrera d'ailleurs comment différentes instructions BASIC peuvent être remplacées par des programmes BASIC.

Vous verrez au chapitre 4.5 qu'il y a un autre système numérique qui joue un grand rôle en informatique. Il s'agit du système hexadécimal. Ce système emploie comme base le nombre 16. Il nécessite donc l'emploi de 16 caractères pour chacun de ses chiffres. Pour représenter les chiffres qui manquent dans le système décimal, on emploie des lettres de sorte que la suite de nombres décimale que nous connaissons bien

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

devient en hexadecimal :

```
1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14
```

La conversion de nombres hexadécimaux en nombres décimaux et inversement se fait selon le même principe que la conversion de nombres binaires. Par exemple :

```
3 F 0 B  =  3*16^3 + 15*16^2 + 0*16^1 + 11*16^0
```

Bien entendu, il est beaucoup plus simple d'entrer en mode direct :

```
PRINT &FF
```

Vous obtenez alors comme résultat le nombre décimal 255.

Notez à cet égard que cette fonction ne permet de convertir que des nombres hexadécimaux inférieurs à 7FFF. Pour les valeurs supérieures, 65536 sera retranché du résultat. Si vous voulez donc convertir des nombres plus grands, vous devez ajouter 65536 au résultat.

Vous pouvez utiliser la fonction HEX$ pour convertir des nombres décimaux en nombres hexadécimaux. Ici aussi, il y a une limite (voyez le manuel d'utilisation). Vous devez vous demander par vous-même quelle est la puissance de 16 la plus élevée qui puisse être contenue dans le nombre décimal, puis former le reste, etc...

Nous nous abstiendrons ici de vous présenter d'autres exemples. Nous vous conseillons toutefois vivement d'effectuer par vous-même un certain nombre de conversions. Et n'oubliez pas que votre CPC vous permet de contrôler très facilement les résultats obtenus.

4.5. Bits et Octets

Le chiffre d'un nombre binaire, c'est-à-dire 0 ou 1 est appelé bit (BInary digiT). Un bit est la plus petite unité d'information que puisse contenir un ordinateur. Pour une valeur de 1 on parlera d'un bit mis et pour une valeur de 0 d'un bit annulé. Comme votre CPC possède un processeur 8 bits, une case mémoire ne peut comporter que des valeurs décimales dont l'équivalent en système binaire ne nécessite pas plus de 8 chiffres (8 bits). Comme 2^8 = 256, le nombre décimal 255 est le plus grand nombre pouvant être représenté avec 8 chiffres binaires :

```
255 = 1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+1*2^0
```

8 chiffres binaires vous permettent donc de représenter des nombres décimaux compris entre 0 et 255. Même si 8 chiffres binaires ne sont pas nécessaires pour représenter des nombres plus petits, on travaille toujours avec 8 chiffres, par exemple 01010101 pour le nombre décimal 85. Un groupe de huit bits ou huit chiffres binaires est appelé octet.

Comme votre CPC possède au total une mémoire de deux fois 65536 (2^16) octets, il ne peut atteindre ces différentes cases mémoire qu'en utilisant deux octets pour leur adressage. C'est pourquoi, en langage machine, les cases mémoires (adresses) sont toujours codées avec un octet fort et un octet faible. Ces deux octets sont placés dans deux cases mémoire différentes. Vous pouvez calculer l'octet fort en divisant l'adresse de la case mémoire correspondante par 256. Un exemple illustrera le maniement des octets faible et fort :

Supposons que vous vouliez calculer les octets fort et faible correspondant à l'adresse 49159, vous devriez effectuer le calcul suivant :

```
49159/256 = 192 , reste 7
```

L'octet fort recevra ainsi la valeur 192 et l'octet faible la valeur 7. Si vous appelez dans un programme une case mémoire (par exemple avec POKE), c'est l'ordinateur qui se charge lui-même de convertir cette adresse en un octet faible et un octet fort.

La plus grande adresse possible, 65535, s'explique tout simplement par le fait que l'octet fort et l'octet faible peuvent chacun atteindre au maximum la valeur 255 :

```
255 * 256  +  255   =   65535
```

Mais peut-être vous êtes-vous déjà demandé quel rôle joue ici le système hexadécimal. C'est tout simple. L'avantage du système hexadécimal est qu'il permet de représenter un octet au moyen de deux chiffres. Cela réduit notamment de façon considérable le temps nécessaire pour entrer des programmes machine ou des données. C'est ainsi par exemple que le nombre 255 devient FF. Pour déterminer la valeur hexadécimale d'un octet, vous n'avez plus qu'à compter avec un exposant maximum de 1 pour la base 16.

Cela permet également de convertir très facilement les nombres binaires en nombres décimaux, par exemple :

```
0011 1010 (bin.) = 3A (hexa) = 3*16^1 + 10*16^0 = 58 (déc.)
```

Vous voyez que le 'détour' par les nombres hexadécimaux vaut le coup car il est très facile de manier 16^1 et 16^0. La conversion d'un nombre binaire de 8 chiffres en un nombre hexadécimal est également très simple puisque chaque moitié du nombre binaire est convertie en un nombre hexadécimal.

4.6. La mémoire du CPC

Si vous consultez le plan d'organisation de la mémoire qui vous est donné dans votre manuel d'utilisation, vous constaterez que le CPC 6128 possède 128 K de RAM et 48 K de ROM. Alors que la RAM (Random Access Memory) est une unité de mémoire qui peut être aussi bien lue qu'écrite, la ROM (Read Only Memory) ne peut être que lue.

La mémoire RAM est en général utilisée pour le stockage des programmes BASIC et des variables. Cette zone s'étend de l'adresse 368 (&170) à l'adresse 43903 (&AB7F). Elle a donc une taille de 43536 octets. La fonction HIMEM vous permet de vous faire indiquer la plus haute adresse occupée par le BASIC. Si vous entrez dans l'ordinateur :

```
PRINT HIMEM
```

vous obtiendrez en sortie sur l'écran la valeur 43903. Ce nombre représente la valeur standard. L'instruction :

```
MEMORY (Adresse)
```

vous permet d'abaisser la zone mémoire disponible pour le BASIC. Cela n'a cependant d'intérêt que si vous avez besoin de place mémoire supplémentaire pour vos programmes machine.

Les 64 K de RAM supplémentaires du CPC 6128 ne peuvent être utilisés à partir du BASIC que si vous lancez le programme BANKMAN qui se trouve sur la première face du paquet de programmes utilitaires. Aux chapitres 6.11, 6.12 et 8.3, nous vous expliquons en détail comment utiliser la RAM supplémentaire.

La ROM se compose du système d'exploitation (zone d'adresses 0 à 16383), de l'interpréteur BASIC (zone d'adresses 49152 à 65535) et de la ROM disquette (également zone d'adresses 49152 à 65535).

Vous vous demandez certainement maintenant comment est réalisée la double occupation de la mémoire. Sans tenir compte de la RAM supplémentaire de 64 K, on obtient déjà une capacité mémoire de 112 K (64 K de RAM et 48 K de ROM) alors que le processeur 8 bits ne peut adresser que 65536 cases mémoire (64 K). Il faut se représenter les choses de la manière suivante: le processeur peut choisir, sélectionner, grâce à une sorte de commutateur, s'il veut utiliser la ROM disquette ou la ROM BASIC (dans la zone haute de la mémoire) ou bien la ROM système d'exploitation ou la RAM (dans la zone basse de la mémoire). C'est le même principe qui est utilisé lorsque vous pouvez accéder à la seconde RAM de 64 K grâce au programme BANKMAN.

Il n'y a plus alors dans ce cas commutation entre RAM et ROM mais entre RAM et RAM.

Vous trouverez à la page suivante un schéma récapitulant l'organisation de la mémoire pour les 64 premiers K :

4.7. Le stockage d'une ligne BASIC dans la mémoire

Lorsque vous écrivez une ligne BASIC dans la mémoire, la RAM est automatiquement sélectionnée, en commençant par la case mémoire 368 ou &170. Nous allons essayer de vous expliquer, dans ce chapitre, comment s'effectue le stockage en mémoire d'une ligne BASIC.

Examinons le programme de deux lignes suivant :

```
1 a=5
2 PRINT a
```

Nous allons prendre l'exemple de ce petit programme pour vous montrer comment le BASIC est placé dans la mémoire. Le

mieux pour cela serait que vous réinitialisiez votre ordinateur avec CONTROL, SHIFT et ESC puis que vous entriez les instructions POKE qui vous sont données plus bas. Nous allons donc POKEr le programme directement dans la mémoire et les cases mémoire correspondantes contiendront les mêmes valeurs que si nous avions entré le programme suivant la manière habituelle. Après que vous aurez entré toutes les instructions POKE, entrez simplement LIST et appuyez sur la touche RETURN. Vous obtiendrez sur l'écran le programme que nous vous avons présenté plus haut. Commençons :

Ecrivons d'abord dans la mémoire la longueur de la première ligne :

```
POKE 368,12
POKE 369,0
```

La première instruction POKE écrit dans la mémoire l'octet faible de la longueur de la ligne alors que la seconde instruction POKE écrit l'octet fort. Si vous ne vous rappelez plus de la signification des termes octet fort et octet faible, reportez-vous au chapitre 4.5.

La longueur de ligne correspond au nombre total de cases mémoire devant être occupées par la ligne BASIC. Votre CPC a besoin de cette indication pour pouvoir calculer l'adresse de départ de la prochaine ligne BASIC.

La prochaine information concerne le numéro de ligne :

```
POKE 370,1
POKE 371,0
```

On place à nouveau les deux octets dans l'ordre octet faible-octet fort. On utilise également deux octets pour le numéro de ligne car la numérotation de lignes peut aller jusqu'à 65535 et non uniquement jusqu'à 255.

On indique ensuite que la variable "a" est une variable numérique :

```
POKE 372,13
```

Si nous avions choisi une variable de texte, nous n'aurions pas ici la valeur 13 mais la valeur 3.

La prochaine valeur à entrer concerne la longueur du nom de variable. La longueur effective doit être augmentée de 4 de sorte que nous obtenons 1+4=5 :

```
POKE 373,5
```

Nous entrons ensuite la valeur 0. Ce zéro sert uniquement comme marque de séparation. Il n'a pas d'autre signification :

```
POKE 374,0
```

Nous en arrivons maintenant au nom de variable. La valeur correspondante se calcule de la façon suivante: valeur ASCII de la dernière lettre + 128, soit 97 + 128 = 225. Pour les noms de variable de plus d'une lettre, les lettres précédant la dernière sont stockées sous leur valeur ASCII normale. Nous entrons donc :

```
POKE 375,225
```

Pour le signe égale qui suit maintenant, nous utilisons une valeur token. La notion de token vous est expliquée au chapitre 4.8. Pour le moment, nous vous demandons simplement de nous croire si nous affirmons qu'il faut entrer :

```
POKE 376,239
```

La prochaine instruction POKE concerne la longueur de la variable. Nous devons indiquer à l'ordinateur que la valeur 5 est inférieure à 256 et qu'elle ne comporte pas de chiffres décimaux. Nous entrons :

```
POKE 377,25
```

Si la valeur de la variable était supérieure à 255, nous devrions écrire dans cette case mémoire la valeur 26 et utiliser pour la

valeur de la variable un octet faible et un octet fort. Pour notre exemple, un octet suffit, de sorte que nous pouvons écrire simplement :

```
POKE 378,5
```

Entrons encore un zéro de séparation :

```
POKE 379,0
```

et nous pourrons maintenant passer à la seconde ligne BASIC.

Nous devons d'abord placer dans quatre adresses les valeurs indiquant la longueur de la ligne et le numéro de ligne :

```
POKE 380,7
POKE 381,0
POKE 382,2
POKE 383,0
```

Nous en arrivons ensuite directement à l'instruction PRINT. La valeur correspondante est un token, comme la valeur représentant plus haut le signe égale. Mais nous n'expliquerons les tokens qu'au chapitre suivant. Nous entrons :

```
POKE 384,191
```

et il ne nous reste plus à écrire dans la mémoire que les valeurs ASCII pour un espace et un 'a' :

```
POKE 385,32
POKE 386,97
```

Les deux lignes BASIC sont maintenant stockées dans la mémoire et vous pouvez le contrôler en examinant le programme avec LIST.

Toute ligne BASIC est en principe de structure semblable à celle de la seconde ligne de programme. Ce n'est que lorsqu'il s'agit de définir des variables que les choses se compliquent un peu. Mais même dans ce cas, on procède fondamentalement comme

nous l'avons décrit ci-dessus. Et si vous rencontrez des problèmes, n'oubliez jamais que vous pouvez lire avec PEEK n'importe quelle case mémoire. Vous pouvez donc parfaitement entrer un programme de la manière habituelle et 'regarder' ensuite quelles sont les valeurs des différentes adresses dans lesquelles le programme est stocké. Il ne vous reste plus alors qu'à trouver vous-même le pourquoi de ces valeurs.

4.8. Les tokens

Le token, mot anglais qui signifie signe, est une valeur, un code, attribué à une instruction BASIC ou à un opérateur logique ou arithmétique. Alors que le texte PRINT occupe normalement cinq octets en mémoire, puisque ce nom se compose de cinq lettres, le token de l'instruction PRINT n'occupe qu'un octet. Les tokens ont donc été développé essentiellement pour gagner de la place en mémoire.

Vous vous demandez peut-être pourquoi, au chapitre précédent, nous n'avons pas pris pour le signe égale la valeur ASCII correspondante mais un token. La raison en est que cela permet d'indiquer directement à l'ordinateur que ce signe égale ne fait pas partie du nom de variable. Il en va de même pour les tokens des instructions BASIC. Les tokens permettent en effet à votre CPC de distinguer plus facilement les instructions des variables.

Pour que vous puissiez travailler avec les tokens, nous vous donnons dans ce chapitre une liste de tous les tokens se composant d'un seul octet. Bien entendu, vous pouvez aussi tester avec un programme tel que le suivant toutes les valeurs de tokens possibles :

```
1 REM
10 INPUT t
20 POKE 372,t
30 LIST 1
```

Si vous entrez par exemple la valeur de token 191, vous obtenez sur l'écran :

```
1 PRINT
```

car la case mémoire 372 occupée auparavant par l'instruction REM a été modifiée avec POKE 372,191.

Le chapitre 5.5.5 vous indique par ailleurs une possibilité intéressante pour employer les tokens pour la protection contre la copie des programmes. Cette méthode utilise notamment les valeurs de token non-utilisées, 226, 232 et 233.

❒ **Mais venons-en à notre liste de tokens :**

Instruction	Valeur	Instruction	Valeur
AFTER	128	AUTO	129
BORDER	130	CALL	131
CAT	132	CHAIN	133
CLEAR	134	CLG	135
CLOSEIN	136	CLOSEOUT	137
CLS	138	CONT	139
DATA	140	DEF	141
DEFINT	142	DEFREAL	143
DEFSTR	144	DEG	145
DELETE	146	DIM	147
DRAW	148	DRAWR	149
EDIT	150	ELSE	151
END	152	ENT	153
ENV	154	ERASE	155
ERROR	156	EVERY	157
FOR	158	GOSUB	159
GOTO	160	IF	161
INK	162	INPUT	163
KEY	164	LET	165
LINE	166	LIST	167
LOAD	168	LOCATE	169
MEMORY	170	MERGE	171
MID	172	MODE	173
MOVE	174	MOVER	175

NEXT	176	NEW	177
ON	178	ON BREAK	179
ON ERROR GOTO	180	ON SQ	181
OPENIN	182	OPENOUT	183
ORIGIN	184	OUT	185
PAPER	186	PEN	187
PLOT	188	PLOTR	189
POKE	190	PRINT	191
CHR$(39)	192	RAD	193
RANDOMIZE	194	READ	195
RELEASE	196	REM	197
RENUM	198	RESTORE	199
RESUME	200	RETURN	201
RUN	202	SAVE	203
SOUND	204	SPEED	205
STOP	206	SYMBOL	207
TAG	208	TAGOFF	209
TROFF	210	TRON	211
WAIT	212	WEND	213
WHILE	214	WIDTH	215
WINDOW	216	WRITE	217
ZONE	218	DI	219
EI	220	FILL	221
GRAPHICS	222	MASK	223
FRAME	224	CURSOR	225
ERL	227	FN	228
SPC	229	STEP	230
SWAP	231	TAB	234
THEN	235	TO	236
USING	237	>	238
=	239	>=	240
<	241	<>	242
<=	243	+	244
-	245	*	246
/	247	^	248
CHR$(92)	249	AND	250
MOD	251	OR	252
XOR	253	NOT	254
ABS	255		

5. Programmation avancée en Basic

5.1. Remarque Préliminaire

Après que le chapitre 3 vous ait fait découvrir les bases du langage de programmation BASIC et que le chapitre 4 vous ait donné quelques notions sur le traitement 'interne' des données, le présent chapitre vous apprendra comment utiliser pleinement toutes les possibilités que recèle le BASIC AMSTRAD. Bien sûr, les éléments du langage présentés au chapitre 3 permettent tout à fait d'écrire des programmes puissants mais cela peut être fait de manière plus élégante si on utilise des instructions supplémentaires.

Pour ce chapitre comme pour les précédents, le principe reste que nous éviterons de répéter les explications de base telles qu'elles figurent dans votre manuel d'utilisation. Nous essaierons plutôt, en partant des connaissances supposées acquises par les 'débutants avancés', de montrer les possibilités offertes par la combinaison de plusieurs instructions et surtout de vous présenter des applications possibles.

5.2. Traitement des chaînes de caractères

5.2.1. Traitement des chaînes de caractères

Nous avons vu au chapitre 3.2 qu'il n'y a pas que des variables numériques mais aussi des variables de texte (appelées variables alphanumériques). Une variable de texte est une chaîne de caractères qui peut contenir jusqu'à 255 caractères quelconques parmi ceux du jeu de caractères présenté dans votre manuel d'utilisation. Vous ne pouvez pas faire avec ces chaînes des calculs comme avec les variables numériques. Seuls l'opérateur arithmétique "+" et les opérateurs de comparaison ">", "<", "=", ">=" et "<=" sont autorisés. Le chapitre 5.2.2 vous explique l'emploi des opérateurs de comparaison. Mais nous

allons d'abord nous intéresser au signe "+" avant d'en venir aux véritables fonctions du traitement des chaînes de caractères.

Le signe plus permet de réunir des chaînes. Le petit programme suivant illustre cette fonction :

```
10 a$="Cher Monsieur"
20 b$=" Meyer"
30 c$=a$+b$
40 PRINT c$
50 END
```

Les deux variables de texte a$ et b$ reçoivent une valeur dans les lignes 10 et 20 puis elles sont réunies en ligne 30 et le résultat est sorti. Notez que nous avons inséré un espace en ligne 20 pour obtenir lors de la sortie :

```
Cher Monsieur Meyer
```

et non pas :

```
Cher MonsieurMeyer
```

Si vous disposez d'une imprimante, vous pouvez utiliser ce principe pour écrire des lettres d'affaires ou de candidature individualisées. Vous pouvez en effet entrer les éléments variables tels que la date, l'adresse, le nom, etc... à travers une instruction INPUT alors que le reste de la lettre sera fixé dans le programme une fois pour toutes. Vous pouvez ensuite faire sortir sur une imprimante connectée à votre ordinateur le texte reconstitué avec le signe plus. Vous comprenez que les lettres écrites de cette façon présentent le double avantage de faire bonne impression et d'être surtout très rapides à écrire.

Mais venons-en aux fonctions de traitement de chaînes proprement dites. En BASIC AMSTRAD, ces fonctions sont tellement pratiques qu'elles vous permettent d'exécuter presque n'importe quelles manipulations de chaînes. Comme nous ne pouvons malheureusement vous présenter dans ce chapitre toutes les applications possibles de ces fonctions, nous vous

invitons une nouvelle fois à vous reporter aux programmes qui vous seront donnés dans les chapitres suivants.

Commençons par la longueur de la chaîne. Entrez d'abord, en mode direct :

```
a$="  3w-alloooooooooooooooooooooooooooo"
PRINT LEN(a$)
```

Vous pouvez bien sûr compter 'à la main' combien de lettres contient cette chaîne. Mais cela va beaucoup plus vite avec la fonction LEN. Cette fonction prend bien sûr en compte tous les caractères, y compris les espaces.

Supposez que, pour dessiner une fonction mathématique, vous vouliez inscrire sur l'axe des x, dans l'angle inférieur droit de l'écran, la plus grande valeur x (voir à ce sujet le chapitre 6.8). Suivant la grandeur de cette valeur, vous parviendriez alors (avec LOCATE) à des positionnements différents. Vous pourriez bien sûr déterminer la grandeur de cette valeur avec de nombreuses instructions IF ... THEN mais dans ce cas l'emploi de la fonction LEN serait certainement plus avantageux et plus élégant. En mode 80 colonnes, l'instruction LOCATE pourrait donc se présenter ainsi :

```
LOCATE 80-LEN(plus grande valeur x),coordonnée y
```

La "plus grande valeur x" doit bien sûr être ici une variable de texte.

Pour déterminer la longueur d'une expression numérique, celle-ci doit tout d'abord être convertie en une chaîne de caractères. Cela est obtenu avec la fonction STR$. Une conversion dans l'autre sens est également possible. On utilise pour cela l'instruction VAL qui vous permet de convertir des chaînes en un nombre. Le résultat est cependant zéro si le premier caractère de la chaîne n'est pas un chiffre. Si le ou les premiers caractères sont bien des chiffres mais qu'apparaissent ensuite des caractères qui n'en sont pas, alors seule la première partie de la chaîne est prise en compte.

Si vous voulez extraire des portions de chaînes d'une chaîne quelconque, vous disposez des fonctions LEFT$, RIGHT$ et MID$.

LEFT$ extrait d'une chaîne une série de caractères consécutifs en commençant par l'extrême gauche de cette chaîne. RIGHT$ est la fonction exactement contraire puisque les caractères sont extraits en comptant à partir de la droite de la chaîne. Le nombre de caractères à extraire est donné par le second argument des fonctions LEFT$ et RIGHT$ alors que le premier argument indique bien sûr la chaîne à traiter. Le programme suivant illustre le maniement de ces fonctions :

```
10 a$="Alain Meunier"
20 PRINT LEFT$(a$,5)
30 PRINT RIGHT$(a$,7
```

Vous obtenez comme résultat :

```
Alain
Meunier
```

Vous verrez au chapitre 5.2.3 comment ces deux fonctions permettent de programmer un message en défilement.

La fonction MID$ est encore plus puissante que LEFT$ et RIGHT$. Elle permet en effet de remplacer aussi bien LEFT$ que RIGHT$. Elle attend trois arguments :

❶ Le nom de la chaîne à traiter

❷ Le numéro du premier caractère à extraire

❸ Le nombre de caractères à extraire

MID$ vous permet donc de retirer d'une chaîne un nombre quelconque de caractères à partir de n'importe quelle position de cette chaîne. Complétez le dernier petit exemple de programme avec :

```
40 PRINT MID$(a$,7,3)
```

et vous obtiendrez comme résultat supplémentaire :

```
Meu
```

MID$ vous permet cependant également de modifier des caractères déterminés d'une chaîne. Si vous entrez par exemple :

```
MID$(a$,3,1)="o"
```

"Alain" deviendra "Aloin", c'est-à-dire que le troisième caractère de la chaîne sera remplacé par un nouveau caractère, "o" en l'occurrence.

La fonction INSTR est une autre instruction très puissante que nous aurons l'occasion d'employer dans la gestion de fichier du chapitre 9. Cette instruction permet de rechercher si une portion de chaîne quelconque se trouve dans une chaîne donnée. INSTR a également trois arguments :

❶ Position, à partir de laquelle la chaîne doit être examinée

❷ Chaîne examinée

❸ Chaîne recherchée

Si vous entrez par exemple en mode direct :

```
a$="Computerbook"
PRINT INSTR(a$,"ter")
```

vous obtiendrez comme résultat la position de la chaîne recherchée, à savoir 6. Un zéro est sorti lorsque la chaîne recherchée n'a pas été trouvée. Comme vous le voyez, l'indication de la position de début de la recherche est facultative lorsque la recherche doit concerner toute la chaîne. La recherche ne donne de résultat positif que si la chaîne recherchée correspond exactement à une portion de la chaîne examinée, en prenant en compte la différence qui existe entre les majuscules et les minuscules.

Lorsque vous voulez convertir une chaîne entièrement en majuscules ou en minuscules, vous pouvez employer les instructions UPPER$ et LOWER$. L'effet de ces instructions est suffisamment simple à constater pour que nous ne nous y étendions pas davantage. Au chapitre suivant nous travaillerons avec l'une de ces instructions.

Les deux dernières instructions qu'il nous reste à décrire permettent de produire des chaînes se composant d'autant de caractères identiques. Avec SPACE$, il s'agira toujours d'espaces alors que STRING$ vous permet d'indiquer un caractère de votre choix. Vous pouvez par exemple tracer une ligne avec :

```
PRINT STRING$(20,"_")
```

SPACE$ peut être utilisé pour différentes choses. Il vous permet par exemple de remplir facilement une chaîne avec des espaces, comme dans les chapitres 6.2.3 ou 9.3. SPACE$ vous permet aussi de positionner une sortie (bien que vous disposiez aussi à cet effet d'instructions puissantes telles que TAB ou SPC) :

```
PRINT SPACE$(5);"c'est la fin du chapitre"
```

5.2.2. Tri

Dans ce chapitre, nous nous intéresserons au tri alphabétique d'un nombre quelconque de chaînes différentes. De telles procédures de tri peuvent être nécessaires dans toutes sortes de domaines d'application. Pensez par exemple au fichier d'une association ou même aux noms dans les annuaires téléphoniques. Mais la routine de tri que nous vous présentons peut également être employée dans des programmes très divers, par exemple pour sortir des résultats triés dans l'ordre alphabétique.

Lorsque vous voulez trier des nombres ou des variables numériques d'après leur grandeur, vous travaillez avec les opérateurs de comparaison "<" et ">". Ces mêmes opérateurs peuvent être utilisés également pour trier des chaînes. Si une

lettre est placée devant une autre dans l'ordre alphabétique, elle sera considérée comme 'inférieure'. Pour les mots de plusieurs lettres, la lettre suivante est automatiquement examinée pour la comparaison si les premières lettres de deux chaînes sont identiques. L'ordinateur effectue donc la comparaison exactement comme vous le faites lorsque vous cherchez un mot dans un dictionnaire ou un nom dans un annuaire.

Il convient cependant de noter une particularité. Du fait des codes utilisés, les majuscules sont toujours considérées comme plus petites que les minuscules. La condition "C" < "c" sera donc toujours remplie. Si vous voulez empêcher que cette règle ne perturbe le tri, vous devez convertir toutes les chaînes à trier en majuscules ou en minuscules, avec UPPER$ ou LOWER$. Vous pouvez alors soumettre les chaînes converties à l'opération de tri, quitte à appliquer le classement ainsi obtenu aux chaînes d'origine.

Dans le programme suivant, les chaînes sont préparées avec l'instruction DATA. Si vous voulez utiliser le programme pour d'autres données (chaînes) de départ, il vous suffit de taper les instructions DATA correspondantes comme lignes de programme. Il vous faudra alors entrer également une nouvelle valeur pour le nombre de chaînes à trier (variable "n"). Le programme suivant présente simplement des données d'exemple, avec n=7.

```
10 REM Tri alphabetique
20 PRINT"Combien de mots doivent etre lus"
30 INPUT"par le programme ";n
40 IF n=0 THEN 330
50 CLS:DIM w$(n),wl$(n)
60 REM Dans ce programme n=7
70 DATA le,bon,ordinateur,pour,vous,c'est,l'AMSTRAD CPC
  6128
200 FOR i=1 TO n
210 READ w$(i)
220 wl$(i)=LOWER$(w$(i))
230 NEXT i
240 FOR i=1 TO n-1
250 FOR j=i+1 TO n
```

```
260 IF wl$(i)<=wl$(j) GOTO 290
270 hi$=wl$(j):wl$(j)=wl$(i):wl$(i)=hi$
280 hi$=w$(j):w$(j)=w$(i):w$(i)=hi$
290 NEXT j
300 PRINT w$(i)
310 NEXT i
320 PRINT w$(n)
330 END
```

❐ Le programme utilise les variables suivantes :

hi$ i j n w$(i) wl$(i)	variable auxiliaire index de comptage index de comptage nombre de chaînes les mots ou les chaînes les chaînes en minuscules

❐ Description du programme :

10	Commentaire.
20-30	Entrée du nombre de chaînes.
40	Saut à la fin du programme s'il n'y a pas de chaînes à lire.
50	L'écran est vidé et de la place mémoire est réservée pour toutes les valeurs des variables w$ et wl$.
60	Commentaire.
70	Préparation des données (les lignes de programme 70 à 190, qui ne sont pas utilisées ici, sont réservées pour des instructions DATA supplémentaires).
200-230	Lecture des données et, en ligne 220, toute chaîne lue est convertie en minuscules (affectation de la variable wl$(i)).

240-320	Les chaînes sont triées dans l'ordre alphabétique et le résultat est sorti. Le tri signifie ici que l'on compare chaque fois deux chaînes entre elles. Pour la comparaison en ligne 260, on utilise les chaînes converties en minuscules, wl$(i). La première chaîne lue, wl$(i), avec i=1, est d'abord comparée avec toutes les autres chaînes wl$(j), avec j=2 ... n. Si lors d'une comparaison on constate que wl$(1) est placé dans l'ordre alphabétique après une autre chaîne wl$(j), alors les deux chaînes comparées sont échangées entre elles dans les lignes 270-280 (l'échange ne concerne par conséquent pas uniquement wl$(i) et wl$(j) en ligne 270 mais aussi w$(i) et w$(j) en ligne 280). Après le parcours de la boucle des lignes 250 à 290, la chaîne placée au début dans l'ordre alphabétique se trouvera en première position (wl$(1), w$(1)). Le résultat est alors sorti directement en ligne 300. La variable de comptage i est ensuite augmentée chaque fois de 1, c'est-à-dire que la chaîne placée en seconde position, wl$(2) sera ensuite comparée avec toutes les autres chaînes (excepté wl$(1)) et échangée s'il y a lieu. On procèdera ensuite de même pour la troisième chaîne, etc... A la fin de l'opération, toutes les comparaisons possibles auront donc été effectuées. Comme la boucle des lignes 240 à 310 a comme valeur finale n 1, la dernière chaîne dans l'ordre alphabétique devra encore être sortie en ligne 320.
330	Fin du programme.

Si vous avez compris la logique de cette opération de tri, alors vous pouvez facilement écrire vous-même une version du programme permettant de trier des valeurs numériques. Vous pouvez bien sûr faire l'économie de la variable wl$ ainsi que des lignes 220 et 270. La variable w$ doit devenir w et hi$ hi. La

comparaison en ligne 260 concernera alors les variables w(i) et w(j). Il ne vous reste plus maintenant qu'à préparer les données correspondantes.

Si vous voulez faire un véritable travail de professionnel, vous pouvez généraliser le programme en insérant au début du programme une interrogation de l'utilisateur pour connaître la nature des données de départ à utiliser. Vous pourrez alors définir les variables utilisées comme des variables de texte ou des variables numériques avec DEFSTR ou DEFREAL. Vous pouvez aussi stocker les données sur disquette pour qu'elles puissent être rechargées à partir de l'extérieur du programme. Le chapitre 8 vous explique comment utiliser le moyen de stockage que constitue la disquette de façon interactive pendant le déroulement d'un programme.

5.2.3. Message défilant

Il vous est certainement déjà arrivé de contempler tel ou tel programme d'animation ou de publicité consistant à faire défiler un message sur l'écran. Les fonctions de traitement de chaînes présentées au chapitre 5.2.1 permettent de programmer un message défilant de ce type sans aucun problème.

Le programme suivant vous permet d'entrer une chaîne ou un texte quelconque comportant au maximum 40 caractères. Ce texte parcourra ensuite l'écran de gauche à droite. La dernière lettre du texte apparaîtra donc d'abord à gauche de l'écran et le texte sortira de l'écran avec sa dernière lettre sur la droite de l'écran. Mais venons-en sans plus tarder au programme :

```
10 REM Message défilant
20 INPUT"Entrez un mot ou un texte ";a$
30 IF LEN(a$)>40 THEN PRINT:PRINT"Chaine trop longue,
  40 caracteres maximum":PRINT:GOTO 20
40 IF LEN(a$)<40 THEN a$=SPACE$((40-
  LEN(a$))/2)+a$+SPACE$((40-LEN(a$))/2)
50 MODE 1
60 FOR i=1 TO 40
70 LOCATE 1,1
```

```
80 PRINT RIGHT$(a$,i)
90 NEXT i
100 FOR i=40 TO 1 STEP -1
110 LOCATE 41-i,1
120 PRINT LEFT$(a$,i)
130 NEXT i
140 END
```

Après un commentaire en ligne 10, le texte est entré en ligne 20. Si ce texte comporte plus de 40 caractères, un commentaire est sorti et on saute à nouveau à l'entrée du texte (ligne 30).

Le texte entré est rempli avec des espaces s'il comprend moins de 40 caractères. On pourra ainsi travailler dans la suite du programme avec une longueur fixe de 40 caractères.

La moitié des espaces nécessaires pour cela est ajoutée à la suite du texte et l'autre moitié devant le texte. Bien entendu, cette répartition n'est pas indispensable mais elle évite simplement que trop de temps ne s'écoule avant qu'on voie quoi que ce soit à l'écran, même pour un petit texte (à cause des espaces).

La ligne 50 vide ensuite l'écran en passant au mode 40 colonnes. Si vous voulez travailler dans un autre mode écran, il vous faut écrire un 2 ou un 8 partout où figure un 4 dans le programme.

Dans la boucle de programme des lignes 60 à 90, on amène lors de chaque parcours de la boucle un caractère de plus à l'écran (en commençant par la droite), à partir de l'angle inférieur gauche de l'écran (voir la ligne 70), jusqu'à ce que la totalité du texte soit visible sur l'écran.

Dans la boucle suivante (lignes 100-130), on procède de façon pratiquement opposée. Le texte est raccourci au fur et à mesure (caractère par caractère) en retranchant un nombre toujours plus élevé de caractères de la fin du texte. Le positionnement avec LOCATE doit également être déplacé ici chaque fois d'une position sur la droite (voir ligne 110). Le premier argument de l'instruction LOCATE est en quelque sorte le complément de la variable de comptage. Plus i devient petit et plus l'expression

41-i augmente, c'est-à-dire que le positionnement évolue horizontalement vers la droite.

Dans ce programme, la ligne la plus élevée est utilisée pour la sortie du message défilant.

Vous pouvez bien sûr choisir d'autres lignes en modifiant les coordonnées y indiquées dans les instructions LOCATE. Si vous voulez par exemple que le texte entré soit sorti successivement dans toutes les lignes de l'écran, vous pouvez compléter ou modifier le programme comme suit :

```
55 FOR j=1 TO 24
70 LOCATE 1,j
110 LOCATE 41-i,j
135 NEXT j
```

Une autre variante de ce programme pourrait consister à sortir un texte différent dans chaque ligne. Si vous intégrez en plus la couleur dans un tel programme (vous 6.5), vous aurez créé ainsi un très beau programme de publicité.

Mais peut-être ne souhaitez-vous pas limiter votre texte à 40 caractères par ligne de texte. Pour qu'un texte plus long reste 'lisible', il faudrait toutefois le faire défiler sur l'écran non pas de gauche à droite mais de droite à gauche (donc en commençant par la première lettre). Faites-en donc l'essai.

Notez par ailleurs que ce programme peut être aussi simplifié ou amélioré. Nous avons dit en effet, au chapitre 5.2.1, que la fonction MID$ peut remplacer les fonctions LEFT$ et RIGHT$. Essayez donc de transformer les deux boucles des lignes 60 à 90 et 100 à 130 en une boucle à paramètres de boucle variables (valeur initiale, valeur finale et pas de progression). Vous pourriez par exemple programmer cette boucle comme un sous-programme et faire appeler ce sous-programme par le programme principal avec des paramètres différents.

Vous voyez donc ici aussi qu'il y a toujours de nombreuses possibilités de modifier et surtout d'améliorer un programme. Et si ces propositions ne vous suffisent pas, alors consultez

donc le chapitre 6.6. Vous y trouverez un programme qui vous permet de faire défiler un texte plus progressivement sur l'écran, c'est-à-dire pixel par pixel.

5.3. Définition de touches et de caractères

5.3.1. La définition des touches

Au chapitre 3, nous avons vu comment l'instruction INKEY permet d'affecter certaines fonctions à des touches déterminées. Le principe était : lorsque telle ou telle touche est appuyée, alors il devra se passer telle ou telle chose dans le déroulement du programme. Les instructions chargées de l'affectation des touches, KEY et KEY DEF, répondent au même souci.

L'instruction KEY permet d'affecter un autre caractère ou une fonction du clavier à une touche. Les tables de codes ASCII ou du jeu de caractères AMSTRAD (voyez le manuel d'utilisation) vous indiquent de quels caractères et fonctions vous disposez.

Pour utiliser l'instruction KEY, vous devez savoir ce qu'est un numéro de caractère d'extension. KEY a en effet les paramètres suivants :

```
KEY numéro de caractère d'extension,chaîne
```

La chaîne est donc affectée à une chaîne désignée par son numéro de caractère d'extension.

Il y a 32 caractères d'extension au total (0-31) qui occupent les valeurs de touche 128 à 159. Ces valeurs de touche ne sont pas identiques aux numéros de touche présentés en haut à droite du clavier. Les valeurs de touche sont présentées dans le chapitre 7 de votre manuel d'utilisation. Les valeurs 128 à 140 correspondent aux chiffres 0 à 9 et au point (bloc numérique de droite) ainsi qu'à la touche ENTER (avec ou sans SHIFT ou CONTROL). Vous pouvez affecter une chaîne directement à ces touches. Si vous entrez par exemple :

```
KEY 137,"LIST"
```

vous verrez ensuite apparaître LIST sur l'écran chaque fois que vous appuierez sur la touche "9" (en haut à droite).

Vous obtenez le même effet en entrant :

```
KEY 9,"LIST"
```

Vous pouvez donc choisir comme argument soit parmi les nombres de 0 à 31, soit parmi les valeurs 128 à 159.

Les caractères d'extension 13 à 31 (valeurs de touche 141 à 159) sont normalement des chaînes vides et ne sont affectés à aucune touche. Si vous affectez une chaîne à un de ces caractères (instruction KEY), vous devez à son tour affecter ce caractère à une touche avec l'instruction KEY DEF. Vous pouvez par exemple affecter également "LIST" à la valeur de touche 150 pour affecter ensuite cette valeur à la touche TAB :

```
KEY 150,"LIST"
KEY DEF 68,1,150
```

Le premier paramètre de l'instruction KEY DEF désigne la touche à laquelle un caractère doit être affecté. Il équivaut au numéro de touche porté sur votre CPC.

Le second paramètre détermine si la fonction de répétition de la touche doit être activée ou désactivée (0=désactivée, 1=activée), c'est-à-dire si la sortie du caractère doit être répétée ou non lorsque la touche est tenue longuement enfoncée. Notez par ailleurs que vous pouvez modifier la vitesse de répétition avec SPEED KEY.

Le troisième paramètre contient la valeur qui doit être sortie lorsque la touche est enfoncée. Cela peut être un caractère ASCII, un caractère d'extension mais aussi une fonction. En ce qui concerne les fonctions, souvenez-vous des codes de commande BASIC évoqués au chapitre 4.3 et énumérés dans votre manuel d'utilisation.

```
KEY DEF Numéro de touche, 0 ou 1,13
```

vous permet par exemple d'affecter la fonction Carriage Return à une touche.

Mais vous pouvez également définir des fonctions du clavier telles que curseur haut avec SHIFT (pour éditer un programme) :

```
KEY DEF Numéro de touche, 0 ou 1,244
```

Au chapitre 6 de votre manuel, vous trouverez une énumération des valeurs correspondantes sous forme hexadécimale. Vous avez vu au chapitre 4.4 comment convertir les nombres hexadécimaux. C'est ainsi que 0D (hexadécimal) = 13 (décimal) ou que F4 (hexadécimal) = 244 (décimal).

KEY DEF vous permet cependant d'affecter également d'autres valeurs à une touche. Ces valeurs seront sorties lorsque cette touche sera enfoncée en même temps que SHIFT ou CONTROL. Ces valeurs représentent les quatrième et cinquième paramètres facultatifs.

Venons-en à l'application pratique des instructions KEY et KEY DEF. Ces instructions peuvent d'une part être employées de belle manière dans des programmes de traitement de texte pour affecter des mots fréquents à des touches déterminées. D'autre part, vous pouvez aussi optimiser votre travail sur l'ordinateur en n'ayant plus à taper les instructions les plus importantes mais en pouvant les obtenir sur l'écran en frappant une simple touche. Les touches numériques de la ligne supérieure du clavier sont celles qui se prêtent le mieux à de telles opérations car vous utilisez probablement le bloc numérique de droite pour taper vos chiffres.

Voici un programme qui vous permet d'affecter n'importe quelles chaînes à 19 touches. Le nombre 19 provient du fait que le programme n'utilise que les numéros de caractères d'extension 141 à 159, l'affectation de chaînes au bloc numérique de droite étant souvent moins pratique. Une fois que vous aurez affecté les textes voulus aux touches que vous

aurez désignées, vous pourrez supprimer le programme avec NEW et vous consacrer à d'autres tâches, par exemple à la programmation.

```
10 REM Affectation de touches pour 19 touches maximum
20 CLS
30 i=i+1
40 PRINT "Affectation"i":"
50 LOCATE 1,5
60 INPUT"Pour quel numero de touche   ";tn
70 PRINT:PRINT:INPUT"Entrez une chaine ";a$
80 LOCATE 1,12:INPUT" avec Carriage Return (o,n) ";x$
90 IF x$<>"o" AND x$<>"n" THEN 80
100 j=140+i
110 IF x$="o" THEN KEY j,a$+CHR$(13) ELSE KEY j,a$
120 KEY DEF tn,0,j
130 LOCATE 1,20:INPUT"Continuer (o,n) ";x$
140 IF x$<>"o" AND x$<>"n" THEN 130
150 IF x$="n" THEN END
160 IF x$="o" AND j<159 THEN 20 ELSE PRINT:PRINT:PRINT
"Vous ne pouvez plus définir que les":PRINT"touches
128-140 (voir manuel)":END
```

❐ Liste de variables :

a$	la chaîne à entrer
i	affectation "actuelle"
j	numéro de caractère d'extension
tn	le numéro de touche
x$	chaîne de réponse (o,n)

Il n'est toutefois pas indispensable que nous donnions ici une description détaillée du programme, étant donné que l'emploi des instructions KEY et KEY DEF a déjà été expliqué en détail plus haut. La ligne 110 présente cependant une nouveauté, la possibilité d'ajouter un Carriage Return (retour de chariot = CHR$(13)) à la chaîne a$ entrée. Cela est surtout intéressant lorsque vous avez entré une instruction pour a$. Le code CHR$(13) entraînera en effet l'exécution immédiate de cette

instruction lorsque vous actionnerez la touche sélectionnée. Il ne sera donc pas nécessaire d'appuyer sur la touche RETURN ou ENTER. C'est ainsi que vous pouvez par exemple affecter l'exécution de l'instruction LIST à la touche TAB, avec :

```
KEY 141,"LIST"+CHR$(13)
KEY DEF 68,0,141
```

Pour le reste, nous vous conseillons d'étudier par vous-même le listing du programme. Vous avez certainement déjà remarqué que la variable i est augmentée lors de chaque 'parcours', c'est-à-dire chaque fois qu'il s'agit de recommencer, et que la variable j est calculée à partir de la valeur de i.

Lorsque vous aurez compris le programme, vous pourrez bien sûr également le modifier ou le compléter. Vous pourriez par exemple intégrer la fonction de répétition ou les touches SHIFT et CONTROL. Il vous suffira alors, pour ainsi dire, de programmer d'autres interrogations (o,n).

5.3.2. Les caractères définis par l'utilisateur

Votre CPC ne vous permet pas seulement de modifier l'affectation des touches mais aussi les caractères eux-mêmes. Pour cela, vous devez savoir que chaque caractère est composé sur l'écran à partir de petits points. Un point de l'écran est également appelé pixel (voir le chapitre 6). Chaque caractère se compose de 64 pixels que vous pouvez 'fixer' (c'est-à-dire rendre visibles) individuellement. Ces 64 points sont disposés dans une matrice de 8x8 que vous pouvez vous représenter comme une grille de 8 lignes sur 8 colonnes.

Pour le travail de définition pratique, le mieux serait que vous preniez une feuille de papier quadrillé et que vous y dessiniez une matrice de 8x8. A vous ensuite de décider quels points vous voulez fixer et quels points vous voulez laisser 'éteints'. Partout où un point devra être fixé, vous placerez un 1 dans la case, sinon un '0'. Une ligne de la grille pourra donc se présenter par exemple ainsi :

```
11000011
0 = couleur de la surface d'écriture
1 = couleur du crayon
```

Comme vous l'indique l'explication des chiffres 0 et 1, une modification éventuelle de la couleur d'écriture ou de la couleur du fond est prise en compte également pour les caractères définis par l'utilisateur (voir également à ce sujet le chapitre 7.5).

Si vous avez encore des problèmes avec la matrice de 8 points sur 8 et avec la définition de vos propres caractères, étudiez un peu la partie du manuel d'utilisation consacrée à ce sujet. Le chapitre 6 du manuel présente en effet tous les caractères de votre CPC dans une matrice de 8x8. Essayez donc de choisir un caractère et de noter les huit nombres binaires correspondant à ce caractère.

Vous obtiendrez par exemple pour le chiffre 1 :

```
0 0 0 1 1 0 0 0
0 0 1 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0
```

Si vous examinez le modèle formé par les points fixés, vous reconnaîtrez nettement le chiffre 1.

Après que vous ayez tout d'abord déterminé les huit valeurs binaires devant définir votre caractère, le reste est assez simple. Vous n'avez plus en effet qu'à réserver de la place pour votre caractère et vous pourrez alors affecter ce caractère à une touche.

L'instruction : SYMBOL AFTER x

indique avec le paramètre x la valeur ASCII du caractère à partir duquel doit commencer la modification. La valeur standard est 240. Si vous choisissez par exemple la valeur 32 pour x, vous pourrez définir vous-même tous les caractères à partir de la valeur ASCII 32. Cependant plus la valeur de x est petite et plus la place mémoire nécessaire sera importante. Comme la mémoire du CPC est toutefois assez importante, il est préférable de choisir une valeur plus petite que nécessaire de façon à ce que vous ayez encore suffisamment de place s'il vous venait à l'idée de modifier ultérieurement encore d'autres caractères. Vous ne devez pas oublier que vous ne pouvez pas employer deux fois l'instruction SYMBOL AFTER car cela supprimerait les caractères déjà définis.

L'affectation elle-même n'est plus alors un problème. Il ne vous faut plus qu'un numéro de caractère, c'est-à-dire le numéro auquel le nouveau caractère devra être affecté (voyez le manuel d'utilisation). Vous pouvez alors redéfinir le caractère de numéro x avec :

```
SYMBOL x,liste des huit (maximum) paramètres de ligne
```

Si vous entrez par exemple :

```
SYMBOL 97,3,3,3,3,3,3,3,3
```

vous obtiendrez une trait horizontal lorsque vous appuuierez sur la touche a (sans SHIFT ni CONTROL) ou lorsque vous entrerez :

```
PRINT CHR$(97)
```

Comme vous le voyez, les paramètres peuvent être indiqués sous forme décimale. Mais vous pouvez aussi (si cela est plus pratique pour vous) employer des nombres binaires ou hexadécimaux. Il vous suffit de marquer ces nombres par un '&'. Vous voyez en tout cas combien sont importants, même pour le programmeur en BASIC, les systèmes numériques que nous vous avons présentés au chapitre 4.4.

Vous aimeriez sans doute savoir maintenant à quoi peuvent donc servir des caractères définis par l'utilisateur. On peut distinguer à cet égard trois domaines d'application :

❶ Les caractères spéciaux techniques ou mathématiques

❷ Les caractères futuristes pour les jeux électroniques

❸ Les caractères accentués français pour le traitement de texte

Comme la troisième application est la plus couramment utilisée, nous vous présentons ci-dessous un programme qui vous déchargera du travail peut-être encore pénible de définition de caractères. Les différents paramètres sont ici indiqués en écriture hexadécimale pour que vous ne perdiez pas l'habitude de la manipulation des systèmes numériques.

Pour les numéros de caractères, nous avons choisi les codes ASCII qui garantissent la bonne restitution des accents français et de la cédille si vous connectez une imprimante à votre ordinateur. C'est d'ailleurs un bon exemple illustrant le rôle du code ASCII pour assurer une transmission correcte des données entre différents appareils (voir le chapitre 4.3).

Comme les touches définies pendant le déroulement du programme ne correspondent pas au clavier normalisé d'une machine à écrire française, nous vous laissons le soin de corriger cela avec l'instruction KEY DEF. Pour le reste, le programme s'explique de lui-même. Vous pourrez l'utiliser chaque fois que vous aurez besoin des accents français.

```
10 REM Caracteres francais speciaux
20 CLS
30 PRINT "Programme de definition des"
40 PRINT TAB(7)"caracteres francais speciaux"
50 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
60 PRINT TAB(9) "Bernd Kowal, 1985"
70 GOSUB 350
80 PRINT"Veuillez actionner lentement l'une"
90 PRINT"apres l'autre les touches suivantes :"
100 PRINT:PRINT"sans SHIFT    : @, \, [, ]"
```

```
110 PRINT:PRINT"et avec SHIFT : |, {, }"
120 SYMBOL AFTER 63
130 REM a accent grave
140 SYMBOL 64,&60,&30,&78,&C,&7C,&CC,&76,&0
150 REM degre
160 SYMBOL 91,&18,&24,&24,&18,&0,&0,&0,&0
170 REM c cedille
180 SYMBOL 92,&0,&0,&3C,&66,&60,&66,&3C,&18
190 REM paragraphe
200 SYMBOL 93,&1E,&30,&38,&6C,&38,&18,&F0,&0
210 REM e accent aigu
220 SYMBOL 123,&C,&18,&3C,&66,&7E,&60,&3C,&0
230 REM u accent grave
240 SYMBOL 124,&30,&18,&66,&66,&66,&66,&3F,&0
250 REM e accent grave
260 SYMBOL 125,&30,&18,&3C,&66,&7E,&60,&3C,&0
270 PRINT:PRINT:PRINT
280 PRINT"Si le dessin des accents ne vous"
290 PRINT"satisfait pas, il vous suffit de"
300 PRINT"modifier les instructions SYMBOL"
310 PRINT"correspondantes."
320 PRINT:PRINT:PRINT
330 PRINT"Fin du programme":END
340 REM SP Attendre
350 LOCATE 7,25
360 PRINT"Frappez une touche S.V.P"
370 x$=INKEY$:IF x$="" THEN 370
380 CLS:RETURN
```

5.4. La technique des fenêtres

5.4.1. La programmation des fenêtres

Votre CPC dispose normalement d'une 'fenêtre' unique que vous pouvez utiliser pour certaines entrées ou sorties. L'instruction WINDOW vous permet de diviser l'écran en jusqu'à huit fenêtres différentes. Chacune de ces fenêtres peut alors être appelée indépendamment pour des entrées ou sorties déterminées. L'instruction WINDOW se présente ainsi :

```
WINDOW#x,g,d,h,b
```

Le paramètre x désigne le numéro de la fenêtre. Si vous négligez la valeur de x, c'est automatiquement la fenêtre 0 qui sera redéfinie. Notez par ailleurs que tous les messages BASIC apparaissent systématiquement dans la fenêtre 0.

Les paramètres g et d déterminent les colonnes limites gauche et droite et les paramètres h et b les lignes limites du haut et du bas de la fenêtre. Suivant le mode écran sélectionné (instruction MODE), vous disposez sur votre écran de 20, 40 ou 80 colonnes sur 25 lignes.

Avec :

```
WINDOW#5,1,40,25,25
```

vous définirez par exemple une fenêtre écran portant le numéro 5 qui ne comprendra que la ligne la plus basse de l'écran.

Si vous voulez maintenant écrire quelque chose dans une fenêtre, entrez simplement, à la suite de PRINT, le symbole dièse et un nombre et la sortie sera déviée sur la fenêtre définie. Avec :

```
PRINT#5,"  voici la fenetre 5"
```

vous obtiendrez donc une sortie dans la fenêtre 5 et avec :

```
CLS#5
```

cette cinquième fenêtre, et uniquement celle-ci, sera à nouveau vidée.

Si vous voulez maintenant placer la fenêtre de sortie standard sur la cinquième fenêtre, alors vous pouvez utiliser l'instruction :

```
WINDOW SWAP 0,5
```

La fenêtre 0 sera alors constituée par la dernière ligne de l'écran et la fenêtre 5 occupera le reste de l'écran.

Il y a toute une série d'instructions différentes que vous pouvez appliquer à une fenêtre déterminée, auxquelles vous pouvez donc affecter avec le dièse un numéro de canal déterminé (canal = appareil ou zone à travers lesquels l'ordinateur sort des informations) :

```
CLS
COPYCHR$
INPUT
LINE INPUT
LIST
LOCATE
PAPER
PEN
POS
PRINT
TAG
TAGOFF
VPOS
WINDOW
WRITE
```

Un certain nombre de ces instructions vous sont sans doute encore inconnues. Mais nous vous les présenterons dans les chapitres suivants de cet ouvrage. Bien entendu, vous pouvez utiliser la plupart de ces instructions sans indication de numéro de canal. Dans ce cas, on suppose automatiquement que c'est le canal ou la fenêtre 0 que vous avez sélectionné.

La définition de la notion de 'canal' donnée ci-dessus laisse entendre que d'autres appareils peuvent être également appelés par l'indication d'un numéro de canal. Il s'agit effectivement du lecteur de disquette intégré (canal 9) et d'une imprimante connectable (canal 8). Nous en parlerons surtout dans les chapitres 8 et 9 de cet ouvrage.

Les fenêtres de l'écran sont généralement utilisée pour améliorer la présentation des programmes. On utilise par

exemple les Windows pour afficher dans une partie déterminée de l'écran des informations dont l'utilisateur a besoin en permanence pour diriger le programme. Ces informations ne sont pas modifiées pendant que les opérations nécessaires pour le déroulement du programme, telles que l'entrée de données ou d'instructions de commande, se poursuivent dans le reste de l'écran.

Vous pouvez cependant parvenir à un effet semblable en utilisant le second groupe de 64 K de RAM. C'est ainsi par exemple que le programme du chapitre 6.12 contient dans une page 'cachée' un guide de l'utilisateur que vous pouvez amener sur l'écran en frappant une touche. Vous trouverez une explication détaillée aux chapitres 6.11 et 6.12.

Mais vous pouvez également utiliser la programmation des fenêtres pour éditer des programmes. Si vous voulez par exemple comparer entre eux deux sections de programme totalement différentes, vous pouvez tout à fait faire sortir les deux sections sur l'écran, dans deux fenêtres côte à côte.

Le domaine des jeux électroniques constitue un autre champ d'application possible de la programmation des fenêtres. C'est ainsi que vous pouvez dessiner dans une fenêtre définie des objets de jeu fixes (une rue, une maison ou autre) de sorte que vous n'ayez pas à redessiner ces objets sans cesse. Le temps ainsi gagné rendra votre jeu certainement plus intéressant car les jeux 'lents' deviennent assez vite ennuyeux.

Indiquons également à ce sujet que vous pouvez aussi définir une fenêtre (appelée fenêtre graphique) pour le graphisme. C'est à cela que sert l'instruction ORIGIN qui vous est présentée au chapitre 6.3.

Voici maintenant quelques programmes pour vous donner quelques idées d'emploi des fenêtres. Mais faites vous-même vos propres essais avec l'instruction WINDOW pour pouvoir ensuite l'employer au mieux dans des programmes plus ambitieux.

5.4.2. Programme de dessin de fenêtres et de masques écran

Vous trouverez dans votre manuel d'utilisation une grille permettant de dessiner des fenêtres écran. Il est très important, avant de commencer un travail de programmation ambitieux, de réfléchir à la disposition de l'écran. Il serait donc préférable que vous sachiez dès le départ quelle partie de l'écran devra être utilisée pour quelle opération du programme.

Le programme suivant vous aidera à planifier la disposition de l'écran. Après que vous ayez indiqué les paramètres correspondant aux différentes fenêtres, celles-ci sont dessinées sur l'écran. Vous pouvez ainsi vous faire une idée de la disposition de l'écran avec les fenêtres ainsi définies et vous pourrez ensuite réemployer les paramètres correspondants dans un autre programme.

Pour que vous ne risquiez pas d'oublier les paramètres entrés, ceux-ci sont sortis après la sortie graphique dans les fenêtres définies. Pour que cette sortie soit possible, encore faut-il que la fenêtre correspondante soit suffisamment grande. Vous constaterez par ailleurs en utilisant ce programme que les dernières instructions WINDOW entrées l'emportent sur les précédentes. Mais venons-en au programme :

```
10 REM Dessin de fenetres
20 MODE 1
30 INPUT"Mode ecran ";bm
40 IF bm<0 OR bm>2 THEN 30
50 MODE bm
60 n=n+1
70 PRINT "Fenetre"n":"
80 PRINT:PRINT:PRINT"Entree des limites":PRINT
90 INPUT"gauche ";l(n)
100 INPUT"droite ";r(n)
110 INPUT"haute ";o(n)
120 INPUT"basse ";u(n)
130 IF n<8 THEN PRINT:PRINT:PRINT:INPUT"Encore une autre
fenetre (o/n) ";a$ ELSE CLS:GOTO 150
140 CLS:IF a$="o" THEN 60
```

```
150 bm=2^bm
160 FOR i=1 TO n
170 xt=l(i):yt=o(i):GOSUB 350:PLOT xg,yg
180 xt=r(i):yt=o(i):GOSUB 350:DRAW xg,yg
190 xt=r(i):yt=u(i):GOSUB 350:DRAW xg,yg
200 xt=l(i):yt=u(i):GOSUB 350:DRAW xg,yg
210 xt=l(i):yt=o(i):GOSUB 350:DRAW xg,yg
220 LOCATE INT(l(i)-1+(r(i)-l(i))/2),INT(o(i)+(u(i)-o(i))/2)
:PRINT i
230 NEXT i
240 x$=INKEY$:IF x$="" THEN 240
250 CLS
260 FOR i=1 TO n
270 WINDOW #i-1,l(i),r(i),o(i),u(i)
280 PRINT#i-1,"gauche ";l(i)
290 PRINT#i-1,"droite ";r(i)
300 PRINT#i-1,"haut ";o(i)
310 PRINT#i-1,"bas ";u(i)
320 NEXT i
330 x$=INKEY$:IF x$="" THEN 330
340 MODE 1:END
350 xg=xt*32/bm-16/bm
360 yt=26-yt:yg=yt*16-8:yt=26-yt
370 RETURN
```

❐ Liste de variables :

a$	Variable de réponse (o,n)
bm	Mode écran
i	Variable de comptage
l(i)	Limite gauche d'une fenêtre
n	Numéro de la fenêtre actuelle
o(i)	Limite supérieure d'une fenêtre
r(i)	Limite droite d'une fenêtre
u(i)	Limite inférieure d'une fenêtre
x$	Valeur de la touche enfoncée
xg	Coordonnée x du curseur graphique
xt	Coordonnée x du curseur de texte
yg	Coordonnée y du curseur graphique
yt	Coordonnée y du curseur de texte

❐ Description du programme :

10-20	Commentaire et vidage de l'écran.
30-50	Entrée d'une valeur pour le mode écran et passage à ce mode.
60	Le numéro "n" actuel est augmenté.
70-120	Entrée des limites pour la fenêtre n.
130-140	Si d'autres fenêtres doivent être définies, on retourne à la ligne 60, sinon l'exécution du programme se poursuit.
150	Modification de la variable bm (0 devient 1, 1 devient 2 et 2 devient 4) pour obtenir un facteur pour les modes 20, 40 ou 80 colonnes (voir le sous-programme en lignes 350 - 370).

160-230	Cette boucle de programme dessine les rectangles correspondant à chaque fenêtre. Les instructions graphiques utilisées ici sont expliquées en détail au chapitre 6.3. La conversion des coordonnées de texte en coordonnées graphiques effectuée dans le sous-programme des lignes 350 370 est expliquée en détail au chapitre 6.6. En ligne 220, le curseur de texte est amené dans le centre de la fenêtre dessinée pour écrire le numéro de la fenêtre dans le rectangle.
240-250	Attente d'une touche et vidage de l'écran.
260-320	Préparation des instructions WINDOW (en commençant par la fenêtre 0) et les paramètres entrés au début du programme sonto sortis dans la fenêtre correspondante.
330-340	Attente d'une touche, passage en mode 40 colonnes et fin du programme.
350-370	Sous-programme de conversion des coordonnées de texte en coordonnées graphiques (voir le chapitre 6.6).

5.4.3. Autres applications

Nous allons essayer, dans ce chapitre, de vous donner encore quelques idées d'application de la technique des fenêtres. Outre une 'window explosion', nous vous montrerons comment représenter en grand format sur l'écran les chiffres 1 et 3. Souvenez-vous à ce propos de la définition des caractères présentée au chapitre 5.3.2. Nous procédons ici pratiquement de la même façon si ce n'est que nous opérons avec tout l'écran et non avec une matrice de 8x8.

L'instruction PAPER utilisée dans les programmes suivants permet de remplir le fond de l'écran d'une couleur. Le chapitre 6.5 vous en dit plus sur cette instruction.

Il ne nous reste plus maintenant qu'à vous souhaiter de bien vous amuser dans vos essais avec WINDOW.

```
10 REM Le chiffre 1
20 MODE 1
30 WINDOW #1,12,15,6,8
40 WINDOW #2,16,23,3,20
50 WINDOW #3,8,31,21,23
60 FOR i=1 TO 3
70 PAPER #i,2
80 CLS #i
90 NEXT i
100 a$=INKEY$:IF a$="" THEN 100 ELSE END
```

```
10 REM Le chiffre 3
20 MODE 1
30 WINDOW #1,8,13,6,8
40 WINDOW #2,12,27,3,5
50 WINDOW #3,24,31,6,11
60 WINDOW #4,16,27,12,14
70 WINDOW #5,24,31,15,20
80 WINDOW #6,12,27,21,23
90 WINDOW #7,8,13,18,20
100 FOR i=1 TO 7
110 PAPER #i,2
120 CLS #i
130 NEXT i
140 a$=INKEY$:IF a$="" THEN 140 ELSE END
```

```
10 REM Window explosion
20 MODE 1
30 gauche=15:droite=25:haut=11:bas=14
40 WINDOW gauche,droite,haut,bas
50 gauche=gauche-1:droite=droite+1:haut=haut-1:bas=bas+1
60 IF haut=0 THEN PAPER 0:END
70 PAPER 3
80 CLS
90 GOTO 40
```

5.5. Trucs et astuces pour le programmeur BASIC

5.5.1. Arrondissements et précision de calcul

La précision de calcul d'un ordinateur se mesure au nombre de chiffres après la virgule (on dit aussi décimales) qui sont conformes à la valeur "exacte". Un ordinateur qui vous donne par exemple la valeur 1.4142 pour SQR(2) n'a pas une très haute précision de calcul. Votre CPC vous donne par exemple pour SQR(2) la valeur 1.41421356, comme vous pouvez aisément le vérifier en entrant :

```
PRINT SQR(2)
```

Les grandeurs réelles ont sur le CPC une précision d'un peu plus de neuf chiffres. Cela signifie que pour les nombres plus grands l'imprécision sera supérieure à 1. Le plus grand nombre qui puisse être stocké correctement est 2^32-1 (=4294967295). Cela tient à la façon dont les variables sont stockées dans la mémoire, comme nous l'avons expliqué au chapitre 4.7. Nous allons illustrer ce principe à nouveau en nous aidant d'un exemple.

Si vous entrez : 2^30+1-2^30

vous obtenez comme résultat la valeur 1.

Mais si vous entrez : 2^32+1-2^32

vous n'obtenez pas comme résultat 1, comme on pourrait s'y attendre, mais 2. L'imprécision est donc de un.

Que ces explications ne vous perturbent pas cependant outre mesure. L'exemple présenté constitue pour ainsi dire un cas limite. Pour la plupart des applications la précision de votre CPC est tout à fait suffisante.

Lorsqu'il s'agit de sortir des résultats, les valeurs arrondies feront pour de nombreuses applications un bien meilleur effet que des nombres avec beaucoup de décimales. Par ailleurs les

valeurs calculées ne cadreront souvent pas, si elles ne sont pas arrondies, avec le format prévu lorsqu'on veut une sortie sous forme de tableau. Comme il y a plusieurs instructions qui permettent d'arrondir de façon adéquate les nombres à nombreuses décimales, nous allons les récapituler ici.

La principale instruction est la fonction ROUND. ROUND vous permet d'arrondir très facilement des nombres ou des valeurs numériques. Par exemple, si vous entrez :

```
PRINT ROUND(5.69228,2)
```

vous obtiendrez la valeur 5.69. Le premier paramètre indique donc la valeur à arrondir alors que le second paramètre détermine le nombre de chiffres auquel il faut arrondir.

Mais vous pouvez aussi arrondir avec l'instruction PRINT USING. Si le format prescrit présente moins de décimales que la valeur devant être sortie formatée, elle est également arrondie, de sorte que 12.278 peut par exemple devenir 12.38 s'il n'y a que deux chiffres après la virgule dans le modèle de format prévu (voyez à ce sujet le chapitre 3.7).

Si, lors de la sortie des valeurs, doit être sortie uniquement la partie du nombre avant le point décimal, alors vous pouvez utiliser l'instruction FIX. Cette instruction détache systématiquement la partie placée avant la virgule alors que l'instruction INT arrondit d'abord au plus proche nombre inférieur. Par exemple :

```
PRINT FIX(-5.6)   donnera le nombre -5,
```

alors que

```
PRINT INT(-5.6)   donnera le nombre -6.
```

Pour les nombres positifs on obtient cependant les mêmes valeurs que pour la fonction FIX.

La fonction INT vous permet également de simuler la fonction ROUND. On utilise pour cela l'équation suivante :

```
z = INT(z*10^s%+0.5)/10^s%
```

z est ici égal au nombre à arrondir et la variable entière s% indique le nombre de chiffres auquel il s'agit d'arrondir. Les chiffres après la virgule qui doivent être arrondis sont ici d'abord ramenés devant le point décimal. On ajoute ensuite 0.5 pour garantir un arrondissement au nombre arrondi véritablement le plus proche du nombre exact. Le nombre ainsi obtenu est alors arrondi avec INT au nombre inférieur le plus proche. La division qui s'ensuit fournit enfin un nombre décimal avec s% chiffres après la virgule.

Nous n'évoquerons maintenant les autres instructions se rapportant à ce problème que brièvement :

La fonction CINT arrondit un nombre ou une variable et le ou la convertit en même temps en format entier. L'éventail de valeurs possibles va par conséquent de -32768 à +32767.

La fonction ABS renvoie la valeur absolue d'un nombre ou d'une variable. Les nombres négatifs deviennent donc positifs.

SGN permet de demander le signe d'une expression numérique. Pour un nombre, on obtient la valeur 1 si le nombre est positif, 0 s'il est égal à zéro et -1 s'il est négatif.

MOD renvoie le reste d'une division, par exemple la valeur 1 pour 13 MOD 4.

Deux autres instructions puissantes permettent de déterminer la valeur maximum ou minimum dans une liste de nombres ou de variables. Il s'agit des fonctions MIN et MAX.

Pour vous montrer comment on peut utiliser ces fonctions également lorsqu'on utilise des variables indexées, voici un petit programme qui calcule non seulement les valeurs maximum et minimum, mais en plus, 'par dessus le marché', la valeur moyenne.

Comme le programme est relativement simple, il n'est pas besoin ici d'explications plus détaillées sur ce programme.

Seules les équations k=a(1) et g=a(1) de la ligne 110 pourraient encore poser problème aux débutants. Il s'agit toutefois simplement d'affecter ici une valeur de départ aux variables k (minimum) et g (maximum) pour pouvoir déterminer les valeurs extrêmes dans les lignes 130 et 140.

Peut-être vous est-il apparu que ce programme traite un problème du même type que le programme de tri (chapitre 4.2.2). Cependant, si vous voulez écrire avec les fonctions MIN et MAX un programme de tri pour valeurs numériques, cela serait certainement pour vous un excellent exercice !

```
10 REM Minimum, maximum et moyenne
20 CLS
30 INPUT"Combien de donnees ";n
40 IF n=0 THEN 190
50 DIM a(n)
60 PRINT:PRINT"Entree de donnees :":PRINT
70 FOR i=1 TO n
80 PRINT "Valeur"i;:INPUT a(i)
90 s=s+a(i)
100 NEXT i:CLS
110 d=s/n:k=a(1):g=a(1)
120 FOR i=2 TO n
130 k=MIN(k,a(i))
140 g=MAX(g,a(i))
150 NEXT i
160 PRINT:PRINT"Minimum =";k
170 PRINT:PRINT"Maximum =";g
180 PRINT:PRINT"Moyenne =";d
190 PRINT:PRINT:PRINT"Fin du programme":END
```

5.5.2. La vitesse de calcul

La vitesse de calcul est une caractéristique de la puissance d'un ordinateur ou d'un programme. En ce qui concerne l'ordinateur, on constate que le CPC fait parfois très bonne figure comparé à d'autres ordinateurs. Mais bien sûr, le plus important pour nous est de savoir comment rendre les programmes le plus rapide possible. Comment doit-on donc

programmer pour obtenir avec l'exécution d'un programme les mêmes résultats en un moindre temps. Disons d'emblée cependant que les programmes du présent ouvrage n'ont pas été écrits en fonction de critères de rapidité d'exécution car cela conduit souvent à réaliser des programmes peu lisibles.

❐ Mais comment mesurer la vitesse d'un programme ?

C'est très simple si vous employez la fonction TIME dans votre programme. TIME indique en effet le temps écoulé depuis l'allumage de l'ordinateur. L'unité utilisée est le trois-centième de seconde.

Il suffit donc d'affecter en début de programme la valeur de TIME à une variable (t par exemple) et d'affecter en fin de programme la valeur de (TIME-t)/300 à une autre variable (z par exemple). z fournira alors la durée du déroulement du programme.

Voici maintenant un exemple de 'mesure de la vitesse' du déroulement d'un programme. Le programme véritable se trouve ici dans un sous-programme appelé en ligne 30.

La mesure est donc réalisée dans le programme principal (il est toutefois également possible de programmer exactement selon le principe inverse).

```
10 REM Vitesse de calcul
20 t=TIME
30 GOSUB 100
40 z=(t-TIME)/300
50 PRINT z;"secondes se sont ecoulees."
60 END
100 FOR i=1 TO 1000
110 NEXT i
120 RETURN
```

Si vous faites tourner ce programme tel quel, vous obtiendrez une valeur de 1.09 secondes. Si vous remplacez les lignes 100 à 120 par :

```
100 FOR i=1 TO 1000:NEXT:RETURN
```

vous obtiendrez une valeur de 1.05 secondes. Le programme est donc devenu un peu plus rapide. Vous pouvez ainsi examiner tous les programmes pour les rendre plus 'rapides'. Vous devez toutefois veiller à ce que le programme plus rapide remplisse toujours la même tâche. Il est d'autre part conseillé de commencer par de petits programmes et par de petites modifications de ces programmes de façon à ce que les raisons d'un déroulement accéléré restent perceptibles.

Pour les programmes d'envergure, vous pouvez parfaitement obtenir un gain important dans le déroulement du programme en prenant en compte une série de considérations mineures. Pour que vous ne vous sentiez pas trop abandonné à vous-même dans cette recherche des 'facteurs de rapidité', voici une présentation de trois procédés permettant d'accélérer l'exécution de vos programmes :

❶ Amélioration du déroulement du programme et des modes de calcul.

Cette méthode est fonction du problème à résoudre. Il n'est donc pas possible de donner des conseils universels dans ce domaine. Toutefois une amélioration de la structure du programme entraînera le plus souvent le gain de temps le plus important. Voyez à ce sujet le chapitre 1.3.

❷ Il faut renoncer autant que possible aux sous-programmes, boucles de programme et branchements.

Cela a cependant un autre inconvénient: la taille des programmes peut en être en effet considérablement accrue jusqu'à ce qu'ils deviennent pratiquement illisibles.

❸ Améliorations de syntaxe et de technique de stockage.

Il s'agit là d'une série de petites considérations mineures qui n'apportent de gain de temps que dans leur ensemble. Ce gain de temps peut toutefois se révéler alors très important dans certains cas.

❐ **Voici quelques conseils :**

a) Renoncez à indiquer la variable d'index pour NEXT.

b) Supprimez de votre programme les espaces inutiles et toutes les instructions REM.

c) Ecrivez autant d'instructions que possible dans une ligne (mais attention: n'oubliez pas qu'on ne peut bien sûr pas sauter au milieu d'une ligne).

d) Utilisez des variables plutôt que des constantes.

e) Dimensionnez vos variables en début de programme.

5.5.3. A la poursuite du hasard

Le CPC possède un générateur de hasard qui vous permet de produire des nombres aléatoires. C'est ainsi que l'instruction RND fournit des nombres aléatoires entre 0.0 et 1.0. Le programme suivant vous permet par exemple de tirer un nombre aléatoire entre 1 et 49 (chiffres du loto) :

```
10 z=INT(RND*49)+1
20 PRINT z
30 END
```

Vous pouvez produire de cette manière des nombres aléatoires dans n'importe quel éventail de nombres. L'équation générale pour la ligne 10 se présente ainsi :

```
z = INT (RND * (limite supérieure - limite inférieure + 1))
+ limite inférieure
```

Si vous choisissez 49 comme limite supérieure et 1 comme limite inférieure, vous obtenez exactement l'équation de l'exemple de programme ci-dessus.

Réinitialisez maintenant votre ordinateur avec CONTROL, SHIFT et ESC et entrez le programme suivant qui produit cinq nombres aléatoires :

```
10 FOR i=1 TO 5
20 PRINT RND
30 NEXT
40 END
```

Comparez maintenant les nombres sur votre écran avec les suivants :

```
0.271940658
0.528612386
0.021330127
0.175138616
0.657773343
```

Vous constaterez que les deux séries de nombres sont identiques. La raison en est que la réinitialisation ou l'allumage de l'ordinateur réinitialisent également la séquence de nombres aléatoires dans laquelle puise votre ordinateur. Les jeux utilisant cette fonction deviendraient donc vite ennuyeux si les actions de ces jeux reposaient toujours sur la même séquence de nombres aléatoires.

Mais comme vous l'imaginez bien, votre CPC vous offre aussi une solution à ce problème. L'instruction à employer est ici :

```
RANDOMIZE x
```

x est une valeur numérique qui représente la valeur de départ pour le générateur de hasard. Si donc vous réinitialisez votre ordinateur et si vous lancez le programme donné plus haut avec l'extension :

```
5 RANDOMIZE 15
```

vous obtenez une autre séquence de nombres aléatoires. Après une nouvelle réinitialisation de votre ordinateur vous obtiendrez cependant à nouveau cette seconde séquence de nombres aléatoires. Vous ne disposez donc toujours pas de nombres 'vraiment' aléatoires. Il nous faut donc un argument pour RANDOMIZE qui soit en règle général différent après chaque réinitialisation de l'ordinateur. C'est la fonction TIME

qui convient ici car on peut partir du principe que le temps écoulé depuis l'allumage de l'ordinateur sera différent lors de chaque lancement du programme. Si vous ajoutez donc la ligne suivante :

```
5 RANDOMIZE TIME
```

dans notre programme, vous obtiendrez une séquence de nombres aléatoires qui ne pourra se répéter deux fois que par l'effet cette fois d'un véritable hasard.

5.5.4. Le traitement des erreurs

Votre CPC vous offre un certain nombre d'aides dans la recherche et l'élimination des erreurs d'un programme. C'est ainsi que l'instruction TRON vous permet de suivre l'exécution d'un programme. Après entrée de cette instruction, le numéro de ligne traité est indiqué en permanence entre crochets. Si en plus vous n'avez qu'une instruction par ligne, cela peut vous fournir une aide considérable. Vous pouvez aussi interrompre momentanément l'exécution d'un programme avec ESC pour demander la valeur de certaines variables.

Pour désactiver cette aide, il vous suffit d'entrer TROFF.

Il est cependant également possible de faire gérer par le programme les erreurs apparaissant lors de son exécution. Si vous utilisez en début de programme l'instruction ON ... ERROR ... GOTO, le programme sautera sans interruption du programme à la ligne indiquée lorsqu'une erreur apparaîtra dans l'exécution du programme. Vous pouvez donc installer à partir de la ligne indiquée une petit 'programme de gestion des erreurs' ou réagir de façon appropriée à l'erreur apparue.

Pour le traitement des erreurs, vous avez cependant besoin d'autres informations. C'est ainsi que la fonction ERL vous permet de savoir en quelle ligne est apparue la dernière erreur alors que ERR vous indique le numéro d'erreur de la dernière erreur apparue. Vous trouverez au chapitre 6 de votre manuel d'utilisation une liste de tous les messages d'erreur. Chaque

message est doté d'un numéro, le numéro d'erreur. Vous pouvez ensuite sortir dans votre gestion d'erreurs un message d'erreur que vous aurez conçu vous-même en fonction du numéro d'erreur. Si l'erreur concerne le système disquette, DERR vous indique de quelle erreur il s'agit. Votre manuel d'utilisation vous fournit également la liste nécessaire. Enfin, ERROR vous permet d'appeler une erreur. L'argument de cette instruction est identique au numéro d'erreur. Le troisième chapitre de votre manuel d'utilisation vous présente un bel exemple vous montrant comment vous pouvez produire vos messages d'erreur propres avec ERROR.

Il est souvent superflu de terminer un programme par le traitement des erreurs. Il peut s'avérer tout à fait suffisant que l'exécution du programme puisse reprendre après la sortie d'une message approprié. C'est à cela que servent les instructions RESUME NEXT et RESUME (numéro de ligne). Avec RESUME NEXT, le programme se poursuit à partir de la ligne suivant la ligne où est apparue l'erreur. RESUME vous permet par contre de déterminer librement en quelle ligne doit se poursuivre l'exécution du programme.

5.5.5. Protection contre les copies

Les programmes professionnels sont maintenant dotés d'une protection contre l'écriture pour empêcher des reproductions non-autorisées. Votre CPC dispose déjà d'une telle protection contre la copie. Lorsque vous sauvegardez un programme avec SAVE, il vous suffit d'ajouter à la suite du nom du programme une virgule et un 'p'. Un programme ainsi sauvegardé ne pourra plus être lancé qu'avec RUN "nom" ou CHAIN "nom". Un tel programme ne peut pas être listé ni sauvegardé. Toute interruption du programme avec ESC a pour effet de le supprimer de la mémoire. Le programme ne peut donc plus qu'être 'exécuté'. C'est pourquoi il est indispensable que l'auteur du programme conserve toujours une version non-protégée du programme pour pouvoir éventuellement y apporter des modifications ou des extensions.

Si ce type de protection de programme ne vous suffit pas ou si vous voulez vous essayer vous-même à la confection de mécanismes de protection, alors souvenez-vous des chapitres 4.7 et 4.8 de cet ouvrage. Nous y avons en effet appris que les valeurs de tokens 226, 232 et 233 ne sont pas définies. Essayez donc d'entrer POKE 372,226 puis de lister ou de lancer un programme. Vous serez étonné du résultat.

Nous avons également vu au chapitre 4.7 comment une ligne BASIC est stockée dans la mémoire. Essayez donc d'imaginer ce qui se passerait si vous placiez au beau milieu de votre programme une instruction POKE qui transforme une ligne REM concernant le Copyright en une ligne PRINT et si la seconde ligne de programme contenait une instruction de programme absolument indispensable pour le bon déroulement du programme.

6. Graphisme

6.1. Introduction

Les possibilités graphiques font certainement partie des caractéristiques les plus intéressantes et les plus fascinantes d'un ordinateur. Dans ce domaine également, votre CPC 6128 appartient à la grande classe si on le compare avec d'autres ordinateurs.

De nombreuses applications dans le domaine du traitement électronique des données sont devenues presque impensables sans un graphisme puissant. Chaque fois que de grandes masses de nombres doivent être compris par l'ordinateur, ils sont représentés graphiquement. Comme disait Napoléon, "un bon croquis vaut mieux qu'un long discours". On pourrait donc tout à fait ranger la présentation visuelle des programmes employés sur le lieu de travail sous le mot d'ordre de 'l'ergonomie', c'est-à-dire de la qualité des conditions de travail. Il importe en effet que non seulement le matériel mais aussi les logiciels soient conçus selon des critères ergonomiques.

Tout ce que votre CPC vous offre dans ce domaine sera présenté en détail dans le présent chapitre. Comme toujours, les explications théoriques seront illustrées par de nombreux exemples et applications. Mais bien entendu, que cela ne vous empêche pas de faire vos propres expériences.

6.2. Résolution graphique et graphisme aléatoire

La résolution graphique fait partie des propriétés d'un ordinateur dont on parle le plus souvent. Comme vous l'avez déjà appris dans le second chapitre de cet ouvrage, cette résolution dépend du mode écran sélectionné. Vous disposez donc au maximum (MODE 2) d'une grille de 640 points horizontalement sur 200 points verticalement. Cela fait 128000 points que vous pouvez faire briller indépendamment les uns

des autres. C'est très certainement suffisant (souvent même largement suffisant) pour pouvoir réaliser de bons graphiques, dessins, diagrammes ou autres jeux.

Si vous avez encore du mal à assimiler la notion de 'résolution', alors essayez simplement de penser à un crayon à mine épaisse et à un crayon à mine fine. Il est évident que vous pouvez faire des dessins plus précis avec un crayon à mine fine car vous obtenez une 'résolution plus élevée'.

On distingue sur tout ordinateur moderne deux modes fondamentaux: un mode 'texte' et un mode 'graphique'. Sur les ordinateurs un peu anciens il faut par exemple entrer une instruction spéciale pour passer d'un mode à l'autre. Sur votre CPC c'est plus simple puisque les instructions graphiques se réfèrent automatiquement au mode graphique.

Le plus simple pour expliquer la différence entre les modes texte et graphique est de se référer à la notion de résolution. Alors que vous disposez en mode texte de 20, 40 ou 80 caractères sur 25 lignes, ce sont en mode graphique 160, 320 ou 640 points horizontalement sur 200 points verticalement. En mode texte vous pouvez donc fixer n'importe quels caractères alors qu'en mode graphique vous ne fixez que des points. Comme en mode texte, il y a un curseur en mode graphique. Ce curseur graphique est cependant invisible, contrairement au curseur de texte. Vous verrez dans les chapitres suivants que cela ne constitue nullement un handicap. Il existe toute une série de différentes instructions graphiques. Le programmeur BASIC non encore 'confirmé' trouvera cela peut-être un peu compliqué. C'est pourquoi nous allons vous montrer maintenant comment on peut déjà obtenir des effets graphiques vraiment intéressants avec les deux instructions de base PLOT et DRAW.

L'instruction PLOT détermine la position du curseur graphique. Elle a au moins deux paramètres. Le premier paramètre détermine la coordonnée x (sens horizontal) et le second paramètre la coordonnée y (sens vertical). Un point ainsi déterminé est affiché directement à l'écran. L'origine des

coordonnées, le point 0,0 se trouve dans l'angle inférieur gauche de l'écran.

Lorsqu'il est question des coordonnées du curseur graphique, on parle souvent de coordonnées de pixel. Un pixel est la plus petite unité représentable sur l'écran. L'écran possède 640 pixels horizontalement sur 400 verticalement.

Si vous voulez donc fixer un point dans le centre de l'écran, vous devez sélectionner 320 comme coordonnée x et 200 comme coordonnée y. Notez que ces coordonnées de pixel restent identiques en mode 0, 1 ou 2. Avec une résolution de 160 points sur 200 (MODE 0), vous devrez donc également entrer PLOT 320,200 pour obtenir un point dans le centre de l'écran.

L'instruction DRAW vous permet de tracer une ligne partant de la position actuelle du curseur graphique. Vous déterminez le but de cette ligne à nouveau par l'indication de coordonnées x et y.

Pour illustrer tout cela, voici une petit programme qui vous montre de premières possibilités de programmation graphique. Il s'appelle tout bonnement 'graphisme aléatoire'. A partir d'un point déterminé au hasard, une ligne est tracée jusqu'à un autre point également déterminé au hasard ... etc... L'opération se répète quatre fois. Une ligne est finalement tracée de la position actuelle du curseur au point de départ.

❑ **Voici comment se présente ce programme :**

```
10 REM Graphisme aléatoire
20 MODE 2
30 x=RND*640:y=RND*400
40 PLOT x,y
50 FOR i=1 TO 4
60 DRAW RND*640,RND*400
70 NEXT i
80 DRAW x,y
```

Lors de chaque lancement du programme vous obtiendrez une image différente. Si vous trouvez que ce programme devient

ennuyeux, alors modifiez simplement la valeur finale de la boucle en ligne 50. Votre dessin deviendra alors 'plus complexe'. Vous pourriez aussi entrer une ligne supplémentaire de programme pour que le programme se déroule 'indéfiniment', par exemple :

```
90 GOTO 30
```

Au bout d'un certain temps, on obtient une image que l'on pourrait appeler 'pelote de laine'.

La fonction RND utilisée ici (voir chapitre 5.5.3) a pour but de donner 'les mêmes chances' à chaque point de l'écran.

6.3. Instructions graphiques puissantes

Dans le dernier chapitre nous avons déjà rencontré deux instructions graphiques de base. Les paramètres utilisés étaient des coordonnées absolues. Absolu signifie ici que la position indiquée par les coordonnées x et y ne dépend que de l'origine des coordonnées. Si vous vous représentez l'écran comme une grille à chaque intersection de laquelle sont affectées une coordonnée x et une coordonnée y, vous pourrez vous repérer directement d'après cette grille lorsque vous indiquez des coordonnées absolues.

Il en va tout autrement lorsqu'on travaille avec des coordonnées relatives. Les coordonnées relatives sont en effet additionnées à la position actuelle du curseur graphique. C'est ainsi que sont calculées les valeurs de coordonnées absolues correspondantes. Le point de référence n'est donc plus l'origine des coordonnées mais la position actuelle du curseur. Si le curseur graphique se trouve par exemple dans l'emplacement 200,200 et si vous voulez avoir un point dans le centre de l'écran, alors vous devrez indiquer la valeur 120 pour la coordonnée x et la valeur 0 pour la coordonnée y.

L'instruction appropriée est ici :

```
PLOTR.
```

Il existe de même une instruction DRAW qui fonctionne avec des coordonnées relatives. C'est :

```
DRAWR.
```

Les instructions graphiques suivantes fonctionnent aussi bien avec des coordonnées absolues qu'avec des coordonnées relatives :

```
MOVE et MOVER
TEST et TESTR
```

L'instruction MOVE x,y sert à déplacer le curseur graphique sans fixer de point comme c'est toujours le cas avec l'instruction PLOT. Alors que MOVE travaille avec des coordonnées absolues, MOVER amène le curseur dans l'emplacement calculé relativement à son ancienne position.

TEST (coordonnées absolues) et TESTR (coordonnées relatives) amènent également le curseur dans l'emplacement indiqué. Mais la fonction TEST ou TESTR indique en même temps le numéro de crayon de couleur qui a été utilisé dans cet emplacement (voyez aussi à ce sujet le chapitre 6.5).

Comme le curseur graphique est invisible, contrairement au curseur de texte, il est souvent nécessaire de demander la position actuelle du curseur. Cela ne pose d'ailleurs aucun problème car vous disposez pour cela des deux fonctions XPOS (pour connaître la coordonnée x) et YPOS (pour connaître la coordonnée y).

Le point de référence pour les coordonnées absolues est, comme nous l'avons dit, l'origine des axes de coordonnées, en bas à gauche de l'écran. Il peut cependant arriver, dans telle ou telle application, qu'on ait intérêt à déplacer cette origine des coordonnées. C'est ce que vous permet l'instruction :

```
ORIGIN x,y.
```

Cette instruction n'atteint cependant sa puissance réelle que lorsqu'on s'en sert pour définir une fenêtre graphique. Outre les

deux indications pour x et y, il faut alors indiquer quatre autres paramètres facultatifs définissant les limites gauche, droite, supérieure et inférieure de la fenêtre. L'instruction ORIGIN fonctionne alors exactement comme l'instruction WINDOW mais uniquement par rapport au mode graphique.

Nous nous sommes jusqu'ici limités à tracer des lignes sur l'écran sous forme d'un trait continu. L'instruction MASK vous permet cependant de dessiner des lignes hâchurées ou pointillées.

Le modèle de point ou de ligne défini avec MASK se répète tous les 8 pixels. On définit donc 8 points consécutifs qui représentent une section de la ligne qui devra être représentée avec DRAW. Cette section sera répétée sans cesse de façon à former une ligne pointillée ou hâchurée.

Le modèle de points est représenté par un nombre binaire. Le chiffre 1 indique qu'un point doit être mis alors que le chiffre 0 indique qu'aucun point ne doit être mis, ce qui crée un vide. Vous obtiendrez par exemple une ligne pointillée en sélectionnant le nombre binaire 01010101. Ce nombre binaire doit être marqué par un '&X' dans l'instruction MASK :

```
MASK &X01010101
```

Dans l'instruction MASK vous pouvez aussi utiliser des nombres décimaux compris entre 0 et 255. Les valeurs binaires ont cependant l'avantage de montrer immédiatement, sans risque d'erreur, comment le modèle de points défini se présente.

6.4. Caractères graphiques

Comme vous avez vu jusqu'ici, votre CPC en connaît un rayon en matière de graphisme. Mais, bien entendu, c'est encore loin d'être tout. Une autre propriété excellente de votre CPC est la possibilité de faire surgir sur l'écran des caractères graphiques au moyen de la fonction CHR$. Essayez par exemple de faire sortir par le programme suivant le jeu de caractères de votre ordinateur :

```
10 FOR i=32 TO 255
20 PRINT i,CHR$(i)
30 NEXT
```

Vous voyez maintenant dans chaque ligne de l'écran (vous pouvez interrompre momentanément le déroulement du programme avec ESC) un nombre et un caractère. Le nombre est ici l'argument de la fonction CHR$ correspondant à chaque caractère sorti.

Vous voyez que vous disposez de caractères graphiques tels que 'petit bonhomme', bombe, etc...

L'emploi de ces caractères est notamment très intéressant dans des programmes de jeu. Bien entendu, vous pouvez aussi définir vous-même chaque caractère comme nous vous l'avons expliqué au chapitre 5.3.2. Mais pourquoi se donner la peine de définir les caractères dont dispose déjà l'ordinateur ?

PRINT CHR$(252) vous permet d'obtenir une bombe sur l'écran. Une bombe, aussi terrible que cela soit dans la vie réelle, possède la désagréable habitude de tomber. La question est donc de savoir comment un tel mouvement peut être représenté à l'écran. Le principe est très simple. LOCATE et PRINT permettent de dessiner une bombe. Immédiatement après on efface à nouveau la bombe dans le même endroit, à l'aide d'une chaîne vide. On dessine ensuite à nouveau une bombe une ligne plus bas, on l'efface à nouveau, etc... On crée ainsi l'illusion du mouvement.

Le programme suivant fait tomber une bombe du bord limite supérieur au bord inférieur de l'écran. Cela se fait au moyen d'une boucle ainsi que d'instructions LOCATE et PRINT. La ligne 110 contient une boucle d'attente qui détermine la vitesse de la chute. Une valeur élevée de g en ligne 40 entraînera une chute plus lente de la bombe car il faudra alors plus de temps pour parcourir la boucle en ligne 110. Notez que la bombe n'est bien sûr supprimée qu'en ligne 120, donc après le parcours de la boucle.

La ligne 70 exploite une autre particularité du CPC: l'instruction FRAME. Cette instruction rend le mouvement de la bombe plus régulier. Cette instruction peut être employée chaque fois que des dessins ou des graphiques se déplacent en glissant sur l'écran.

```
10 REM La bombe choit
20 PRINT"Entree de la vitesse (plus le nombre  "
30 PRINT"est eleve, plus la chute sera lente)  "
40 INPUT g
50 MODE 2
60 FOR i=1 TO 25
70 FRAME
80 LOCATE 40,i
90 PRINT CHR$(252)
100 LOCATE 40,i
110 FOR j=1 TO g:NEXT j
120 PRINT" "
130 NEXT i
```

6.5. La magie des couleurs

Le CPC 6128 est aussi une machine remarquable en ce qui concerne les couleurs. C'est d'ailleurs pourquoi vous trouvez immédiatement à côté du clavier la table des couleurs du CPC qui ne présente pas moins de 27 couleurs. Ces couleurs ne peuvent toutefois être toutes utilisées en même temps. En mode 0 vous pouvez disposer de 16 couleurs différentes, de 4 en mode 1 et de 2 seulement en mode 2.

On pourrait consacrer un chapitre très détaillé aux couleurs et notamment aux effets qu'on peut obtenir grâce aux couleurs. En principe les couleurs peuvent être utilisées dans tous les domaines. Mais avant de faire un usage des couleurs trop généreux, il convient de bien réfléchir au but recherché. Vos yeux vous seront par exemple infiniment reconnaissants si vous ne leur imposez pas de travailler en traitement de texte avec des couleurs trop nombreuses ou pire avec des couleurs clignotantes. Pour de nombreuses applications (qui nécessitent que le regard soit fixé sur l'écran) toute profusion de couleurs

est déplacée. Seuls les jeux ont un besoin vital de couleurs (entre autres). Tel ou tel graphique sera peut-être aussi plus agréable à regarder en couleur. Excepté donc les programmes de jeu, on peut dire que la plupart des programmes se passent fort bien de couleurs ou du moins qu'il sera toujours temps de réfléchir à l'emploi des couleurs dans le programme une fois que tout le reste sera terminé.

Pour ces raisons mais aussi pour que les possesseurs d'un moniteur monochrome vert ne se sentent pas trop frustrés, nous n'avons pas développé ce chapitre sur les couleurs autant que nous l'aurions pu. Notez cependant que votre manuel d'utilisation explique aussi les instructions de couleur de manière assez complète. Voici donc maintenant une présentation brève mais précise des principaux éléments concernant les couleurs.

Chaque fois que vous avez affaire aux couleurs, vous ne devez pas oublier que vous pouvez définir les couleurs du bord de l'écran (BORDER), du fond (PAPER) et du crayon (PEN) indépendamment les unes des autres. L'instruction :

```
BORDER x
```

vous permet de choisir un numéro de couleur directement dans la table des couleurs du CPC. La couleur du bord de l'écran sera alors automatiquement modifiée en conséquence.

Il en va différemment avec les instructions PEN et PAPER. L'instruction PEN vous permet de modifier la couleur du crayon et l'instruction PAPER de modifier la couleur du fond de l'écran mais cela ne se fait pas directement par l'indication d'un numéro de couleur. Avec PEN et PAPER, vous indiquez en effet des numéros correspondant à des couleurs déterminées. Après l'allumage de l'ordinateur, PEN est par exemple fixé à 1 et PAPER à 0. Au numéro 1 correspond le numéro de couleur 24 (jaune clair) et au numéro 0 le numéro de couleur 1 (bleu). Si vous voulez maintenant affecter une autre couleur au numéro 1 pour PEN/PAPER, vous devez utiliser l'instruction INK. Vous pouvez par exemple affecter la couleur blanc brillant au numéro 1 pour PEN/PAPER avec :

```
INK 1,26
```

Le crayon aura alors la couleur blanc brillant. Pour modifier cela à nouveau, vous pouvez entrer une nouvelle instruction INK ou affecter au crayon un autre numéro avec PEN. Vous pouvez bien sûr procéder exactement de même avec PAPER.

La table d'affectation de PAPER/PEN/MODE/INK (chapitre 1 de votre manuel) vous indique quelle couleur est affectée de manière standard à quel numéro PEN/PAPER (cela dépend aussi du mode écran sélectionné).

Ce que nous avons dit jusqu'ici pour PEN et PAPER s'applique également au domaine graphique avec les instructions GRAPHICS PAPER et GRAPHICS PEN.

Vous pouvez définir les couleurs de vos dessins en affectant (comme troisième paramètre) un crayon de couleur aux instructions graphiques PLOT, PLOTR, DRAW, DRAWR, MOVE ou MOVER. Vous pouvez encore doter ces instructions d'un quatrième paramètre appelé mode couleur. Ce paramètre vous permet de combiner la couleur du crayon de couleur utilisé avec les autres couleurs de l'écran. Vous disposez en tout de quatre modes couleur. Pour en connaître le mode d'emploi nous vous invitons à vous reporter à votre manuel d'utilisation et de faire vos propres expériences. Voici maintenant un exemple simple d'emploi de l'opérateur logique XOR (valeur de paramètre 1) :

```
10 CLS
20 MOVE 50,50,,1
30 DRAW 50,300
40 DRAW 500,300
50 DRAW 500,50
60 DRAW 50,50
70 GOTO 20
```

Le quatrième paramètre de l'instruction MOVE provoque le changement de couleur.

Interrompez le programme et entrez :

```
20 MOVE 50,50,,0
```

et supprimez la ligne 70. Vous obtiendrez maintenant, après avoir lancé le programme, un beau rectangle sur l'écran car la valeur de paramètre 0 entraîne un retour au mode couleur normal. Vous pouvez remplir ce rectangle avec une couleur, au moyen de l'instruction FILL, après une nouvelle interruption du programme. Avant cela, vous devez toutefois encore placer le curseur graphique dans le rectangle, par exemple avec : MOVE 100,100.

Vous pouvez alors, avec :

```
FILL 3
```

par exemple remplir ensuite la surface entre les lignes avec la couleur affectée au numéro 3 pour PEN/PAPER.

6.6. Graphisme et texte

De nombreux programmes puissants tirent l'essentiel de leur intérêt de la possibilité de représenter en même temps sur l'écran des dessins et du texte. Pensez par exemple au figures ou aux programmes de jeu. Mais malheureusement le curseur de texte et le curseur graphique ne sont pas identiques. Alors que la position 0,0 du curseur de texte est dans le coin supérieur gauche de l'écran, l'origine du curseur graphique est en bas à gauche de l'écran.

Il est naturellement possible de déplacer l'origine du curseur graphique, avec ORIGIN, par exemple pour l'amener aussi dans l'angle supérieur gauche. Mais le calcul de la position d'un des deux curseurs à partir de la position de l'autre curseur ne peut se faire au moyen d'un facteur unique car, d'une part, les signes des coordonnées x et y ne sont pas les mêmes et, d'autre part, le curseur de texte dépend du mode écran actuel.

Mais, comme nous y sommes habitués avec le BASIC AMSTRAD, il y a une instruction appropriée pour (presque) chaque problème. En l'occurrence il s'agit des instructions TAG

et TAGOFF. Si TAG est activé, le texte suivant sera sorti là où se trouve le curseur graphique. L'angle supérieur gauche du caractère de texte sera alors placé sur le curseur graphique. Avec TAGOFF le curseur de texte est replacé dans la position qu'il occupait avant l'instruction TAG.

En ce qui concerne l'instruction PRINT suivant l'instruction TAG, il convient de veiller à ce que celle-ci se termine par un ';' car sinon les caractères de commande tels que le retour de chariot ou le passage à la ligne seront sortis avec le texte. Il faut d'autre part noter qu'en mode direct TAG est désactivé par l'indication du canal 0.

L'instruction TAG peut être utilisée pour positionner des textes de manière plus précise, c'est-à-dire pixel par pixel. De cette façon cela ne pose plus aucun problème de programmer un message défilant. Cela pourrait se présenter ainsi :

```
10 REM Message defilant
20 CLS
30 TAG
40 FOR i=-128 TO 640
50 MOVE i,200
60 PRINT" Computer";
70 NEXT
80 TAGOFF
```

Après le lancement du programme, le mot 'Computer' se déplacera sur l'écran, pixel par pixel, de gauche à droite. Comme cela dure un certain temps, il est conseillé de choisir en ligne 40 un autre pas (STEP 8 par exemple). La valeur initiale de la boucle, - 128, provient du fait que le mot 'Computer' comporte en mode écran 1 exactement 128 pixels de longueur (16 pixels par lettre). C'est qu'il ne s'agit pas en effet de faire apparaître immédiatement sur l'écran le mot 'Computer' mais seulement en défilement. Il faut également noter qu'en ligne 60 le mot lui-même est précédé d'un espace pour que soient effacées au fur et à mesure les parties écrites devenues inutiles. Si vous voulez maintenant employer les couleurs dans ce programme, alors n'oubliez pas que ce sont les instructions

GRAPHICS PEN et GRAPHICS PAPER qui sont responsables des couleurs pour un texte ainsi sorti sur l'écran.

Il peut cependant également arriver que vous vouliez définir des coordonnées graphiques en fonction de coordonnées de texte. Vous ne pouvez le faire avec l'instruction TAG. Il vous faudrait donc réfléchir au meilleur placement de votre dessin. C'est pourquoi nous vous présentons un programme qui vous évitera ce type de travail. Il vous permet de convertir des coordonnées de texte en coordonnées graphiques mais aussi l'inverse, c'est-à-dire pratiquement de simuler l'instruction TAG. Cela montre bien que les instructions BASIC peuvent être remplacées par un logiciel approprié. Il est cependant toujours préférable de profiter au maximum du jeu d'instructions disponible pour que les programmes restent le plus clairs et les plus simples possible. Cela augmente aussi la vitesse de travail de l'ordinateur.

Dans le programme suivant, la conversion est effectuée après une sélection dans un menu.

Les lignes 180 et 190 sont particulièrement importantes à cet égard. Elles peuvent parfaitement être employées dans tel ou tel sous-programme (sans l'instruction IF bien entendu). Après cette conversion, les valeurs de coordonnées correspondantes sont sorties sous forme de texte et sous forme graphique. Le calcul des coordonnées se fait de telle manière que les coordonnées graphiques calculées soient placées au centre du caractère de texte. Lorsqu'on calcule les coordonnées de texte, cela n'est bien sûr pas possible. Elles recouvrent simplement les coordonnées graphiques.

```
10  REM Conversion de coordonnees texte-graphiques ou graphiques-texte
20  CLS
30  PRINT:PRINT"Conversion des positions du curseur"
40  PRINT:PRINT:PRINT TAB(33)"Entree"
50  PRINT:PRINT:PRINT"Texte-graphiques";TAB(36)"1"
60  PRINT:PRINT"Graphiques-texte";TAB(36)"2"
70  PRINT:PRINT:PRINT:INPUT"Votre choix ";a
80  IF a<1 OR a>2 THEN 30
90  PRINT:PRINT:PRINT:PRINT:INPUT"Mode ecran (0,1 ou 2) ";bm
```

```
100 CLS:IF bm <0 OR bm>2 THEN 90
110 IF a=2 THEN 140
120 INPUT"Coordonnee x (texte) ";xt
130 PRINT:INPUT"Coordonnee y (texte) ";yt:GOTO 160
140 INPUT"Coordonnee x (graphique) ";xg
150 PRINT:INPUT"Coordonnee y (graphique) ";yg
160 PRINT:PRINT:PRINT"cela equivaut a:":PRINT
170 bm=2^bm
180 IF a=1 THEN xg=xt*32/bm-16/bm ELSE xt=INT(xg/32*bm)+1:IF xg=640
THEN xt=20*bm
190 IF a=1 THEN yt=26-yt:yg=yt*16-8:yt=26-yt ELSE yg=400-
yg:yt=INT(yg/16)+1:yg=400-yg
200 IF a=2 THEN 220
210 PRINT:PRINT"x (Graphique) =";xg:PRINT:PRINT"y (Graphique)
=";yg:GOTO 230
220 PRINT:PRINT"x (Texte) =";xt:PRINT:PRINT"y (Texte) =";yt
230 PRINT:PRINT:PRINT:PRINT:PRINT"Frappez une touche S.V.P."
240 a$=INKEY$:IF a$="" THEN 240
250 MODE INT(bm/2)
260 LOCATE xt,yt:PRINT CHR$(143)
270 IF INT(bm/2)=2 THEN f=2 ELSE f=3
280 PLOT xg,yg,f
290 a$=INKEY$:IF a$="" THEN 290 ELSE MODE 1:END
```

❐ Liste de variables :

a	Réponse 'choix du menu'
a$	Renvoie la valeur de la touche enfoncée
bm	Mode écran et facteur
f	Crayon de couleur du point pour la sortie graphique
xg	Coordonnée x "Graphique"
xt	Coordonnée x "Texte"
yg	Coordonnée y "Graphique"
yt	Coordonnée y "Texte"

❐ Description du programme :

10-20	Commentaire et vidage de l'écran.
30-80	Sélection du menu.
90-100	Choix du mode écran.
110-150	Entrée des coordonnées x et y devant être converties (texte et graphisme séparés).
160	Sortie d'un titre pour la sortie de résultats.
170	La variable bm est transformée en facteur (0 devient 1, 1 devient 2, 2 devient 4).
180	Les coordonnées x 'graphique' ou 'texte' sont calculées. Rappelons qu'en mode 1 16 pixels sont mis à disposition pour chaque caractère. Cette valeur est divisée ou multipliée par deux pour les autres modes écran. Lors du calcul des coordonnées de texte il faut tenir compte du cas particulier xg=640. Une instruction particulière empêche ici que le curseur de texte dépasse la limite droite de l'écran.
190	Calcul des coordonnées y 'graphique' et 'texte'. Le mode écran ne joue aucun rôle ici en tant que facteur car un autre mode modifie seulement la division de l'écran en colonnes. Comme le point zéro de la coordonnée y se trouve une fois en haut à gauche et une autre fois en bas à gauche de l'écran, les coordonnées connues sont chaque fois 'inversées' avant et après le calcul, c'est-à-dire que la coordonnée de texte 2 devient la coordonnée 24 pour que la coordonnée graphique puisse ensuite être facilement calculée.
200-220	Sortie des résultats.
230-240	Attente d'une touche avec sortie de commentaire.

250	Le mode écran est modifié en fonction de l'entrée en ligne 90.
260-280	En fonction des coordonnées sélectionnées ou calculées on dessine un caractère de curseur de texte et un point. La couleur du point se distingue ici de la couleur du caractère. Comme en mode 2,la même couleur est affectée au troisième crayon de couleur qu'au premier (cas standard), on choisit dans ce cas f=2 et non f=3.
290	Attente d'une touche, vidage de l'écran et fin du programme.

6.7. Histogrammes et graphiques camembert

De nombreux fabricants de logiciels axent leur publicité sur le fait que leurs programmes représentent les résultats sous forme d'histogrammes ou de graphiques camembert. Ce mode de présentation est bien sûr agréable à l'oeil mais il est aussi tout à fait valable du point de vue 'ergonomique'. De plus la programmation de tels graphiques n'est pas si compliquée que vous vous l'imaginez peut-être, grâce à l'excellent BASIC AMSTRAD.

Nous allons maintenant vous présenter trois programmes amplement commentés que vous pourrez sans problème utiliser comme sous-programmes dans d'autres programmes. Chaque fois que vous obtenez des résultats dans un programme vous pourrez donc les représenter sous une forme graphique sans gros travail de programmation. Vous devrez simplement faire en sorte que les tableaux 'nom$' et 'valeur', qui sont ici remplis au clavier, soient remplis à partir du programme principal.

Le premier programme de ce chapitre construit un histogramme. Suivant la valeur de données entrées (10 au maximum) les histogrammes seront dessinés avec la fonction :

```
CHR$(143)
```

Un histogramme s'étendant du bord inférieur au bord supérieur de l'écran est affecté à la valeur la plus élevée. On procède ensuite de façon similaire pour les autres valeurs. La hauteur de l'écran est donc toujours entièrement exploitée, indépendamment de la grandeur des valeurs de données.

```
10 REM Histogrammes
20 CLS
30 PRINT"Combien de donnees differentes faut-il"
40 INPUT"representer (10 maximum) ";nombre
50 IF nombre>10 OR nombre<2 THEN 20
60 PRINT:PRINT"Vous pouvez entrer maintenant un"
70 PRINT"nom (3 caract. maxi) et une valeur
80 PRINT"numerique pour les donnees":PRINT
90 FOR i=1 TO nombre
100 INPUT"Nom ";nom$(i)
110 IF LEN(nom$(i))>3 THEN nom$(i)=LEFT$(nom$(i),3)
120 LOCATE 15,i+7:INPUT"Valeur ";valeur(i)
130 IF valeur(i)>valeurmax THEN valeurmax=valeur(i)
140 somme=somme+valeur(i)
150 NEXT i
160 FOR i=1 TO nombre
170 valeurrel(i)=valeur(i)*100/valeurmax
180 NEXT i
190 CLS
200 PLOT 10,17:DRAW 10,380
210 PLOT 10,17:DRAW 640,17
220 LOCATE 1,1:PRINT"%"
230 FOR i=2 TO nombre*4 STEP 4
240 j=j+1
250 IF j=10 AND LEN(nom$(j))=3 THEN TAG:MOVE 585,14:PRINT
nom$(j);:TAGOFF:GOTO 280
260 IF LEN(nom$(j))=1 THEN LOCATE i+1,25 ELSE LOCATE i,25
270 PRINT nom$(j)
280 x=ROUND(22*valeurrel(j)/100)
290 FOR l=24 TO 24-x STEP -1
300 LOCATE i+1,l:PRINT CHR$(143)
310 NEXT l
320 LOCATE i-1,23-x:PRINT ROUND(valeur(j)*100/somme)
330 NEXT i
340 a$=INKEY$:IF a$="" THEN 340 ELSE :CLS:END
```

❐ Liste de variables :

a$	Chaîne renvoyant la touche enfoncée
nombre	Nombre de valeurs
i	Index de comptage
j	Index de comptage
l	Index de comptage
nom$(i)	Désignation des données
somme	Somme des valeurs de données
valeur(i)	Les valeurs de données
valeurmax	La valeur maximum
valeurrel(i)	Les valeurs relatives par rapport à la valeur maximum
x	Variable auxiliaire pour le dessin des histogrammes

❐ Description du programme :

10-20	Commentaire et vidage de l'écran.
30-50	Affectation de la variable "nombre".
60-80	Sortie de commentaire.
90-150	Entrée des données et des noms correspondants. Si un nom comporte plus de trois caractères, seuls les trois premiers sont pris en compte (ligne 110). Les lignes 130-140 calculent directement la valeur maximum et la somme des valeurs.
160-180	Cette boucle de programme calcule les valeurs relatives par rapport à la valeur maximum. Il s'agit ici d'un calcul de pourcentage avec la valeur maximum (100%) comme valeur de base (pourcentage = valeur de pourcentage * 100 / valeur de base).
190-210	Vidage de l'écran et dessin des axes de coordonnées
220	Ecriture de l'axe des y.

230-330	Cette boucle de programme écrit l'axe des x et dessine les histogrammes. L'index de comptage i de la boucle de programme indique ici la position du curseur (de texte) dans le sens horizontal. Le premier histogramme est donc dessiné en seconde position, le second en sixième position, etc... L'index j permet de marquer les variables "nom$", "valeur" et "valeurrel". Il est augmenté de 1 lors de chaque parcours (ligne 240). En lignes 250-270, le nom entré est inscrit sur l'axe des x. La ligne 260 fait en sorte qu'un nom ne se composant que d'un seul caractère soit présenté directement en dessous de l'histogramme. Comme avec l'entrée de 10 noms le dernier nom (s'il se composait de trois caractères) entraînerait un scrolling de l'écran, sa sortie est décalée en ligne 250 d'un pixel vers la gauche au moyen de la fonction TAG. La ligne 280 calcule la variable auxiliaire x en fonction de "valeurrel(j)". Pour la valeur maximum, "valeurrel" vaut 100. x atteint alors également son maximum, c'est-à-dire 22. Du fait des arrondissements nécessaires (on a besoin de nombres entiers pour x), il peut arriver que des histogrammes de taille identique soient affectés à des valeurs légèrement différentes. Si vous examinez de plus près l'équation en ligne 280, vous constaterez qu'elle est issue du calcul de pourcentage, à savoir : Valeur de pourcentage = valeur de base * pourcentage / 100. x est ici la valeur de pourcentage, 22 la valeur de base et "valeurrel" le pourcentage. En lignes 290-310 sont maintenant dessinés les histogrammes avec la fonction CHR$(143). Notez que la valeur finale de la boucle en ligne 290 est déterminée en fonction de la variable auxiliaire x. Avant la fin d'un parcours de boucle en ligne 330, les valeurs de pourcentage des différents histogrammes (avec "somme" comme valeur de base) sont sorties en ligne 320. Nous avons renoncé à indiquer les décimales car la sortie est positionnée en dehors des histogrammes correspondants. Si vous voulez supprimer ces informations, vous pouvez supprimer la ligne 320 et remplacer en ligne 280 la valeur de base par 23.
340	Boucle d'attente (pour empêcher un scrolling) et fin du programme.

Lorsque vous connaîtrez mieux les instructions graphiques de votre ordinateur, vous penserez certainement, à juste titre, que les histogrammes pourraient être représentés de manière plus belle et plus précise.

Il nous faut pour cela renoncer à l'emploi de CHR$ et travailler plutôt avec les instructions graphiques MOVE, PLOT, DRAW et FILL. Comme nous n'avons plus en effet dans ce cas 25 caractères mais 200 points que nous pouvons fixer individuellement, une représentation plus précise des histogrammes devient possible. De plus, les histogrammes peuvent être aussi représentés en couleur et même en trois dimensions sans que cela pose trop de problèmes de programmation. Voici donc comment se présente la version améliorée du programme (nous ne vous présentons pas à nouveau les lignes 10 à 220 car elles n'ont pas été modifiées) :

```
230 FOR i=2 TO nombre*4 STEP 4
240 j=j+1
250 IF j=10 AND LEN(nom$(j))=3 THEN TAG:MOVE 585,14:PRINT
nom$(j);:GOTO 300
260 IF LEN(nom$(j))=1 THEN LOCATE i+1,25 ELSE LOCATE i,25
270 PRINT nom$(j)
280 NEXT i
290 TAG
300 j=0
310 FOR i=29 TO 605 STEP 64
320 j=j+1
330 IF j>nombre THEN 420
340 x=ROUND(376*valeurrel(j)/100)
350 IF x<18 THEN x=18
360 PLOT i,17:DRAW i,x:DRAW i+16,x:DRAW i+16,17
370 PLOT i,x:DRAW i+6,x+6:DRAW i+22,x+6:DRAW i+16,x:MOVE
i+22,x+6:DRAW i+22,17+6:DRAW i+16,17
380 MOVE i+18,20:FILL 3:MOVE i+14,18:FILL 3:MOVE i+16,x+3:FILL 3
390 MOVE i-16,x+22:PRINT ROUND(valeur(j)*100/somme,0);
400 NEXT i
410 TAGOFF
420 a$=INKEY$:IF a$="" THEN 420 ELSE :CLS:END
```

Comme vous le voyez, nous n'avons pas employé de variables supplémentaires. Les lignes de programme 230 à 270 n'ont pas non plus été modifiées si l'on excepte l'instruction de saut en ligne 250.

Cependant, la boucle ouverte en ligne 230 est fermée dès la ligne 280 car, avec les 640 pixels disponibles, l'index i, qui détermine ici aussi la position horizontale, doit prendre d'autres valeurs qu'avec 40 caractères. Les lignes 230-280 servent donc uniquement à écrire sur l'axe des x. L'instruction TAG est alors fixée et j reçoit à nouveau la valeur initiale 0 pour que puisse être ensuite formée une nouvelle boucle pour i à partir de la ligne 310. Les valeurs initiale et finale ainsi que le pas de la boucle sont organisées de telle façon que l'histogramme ne soit pas dessiné directement au-dessus du nom correspondant. La ligne 330 examine alors s'il existe encore d'autres valeurs j car la variable "nombre" n'a pas été reprise comme valeur finale de la boucle. La variable auxiliaire x est ensuite calculée avec la valeur de base modifiée 376 (puisqu'on dispose maintenant de 400 pixels dans le sens de la verticale). Si x était inférieur à 18, x serait fixé à 18 pour qu'on puisse au moins dessiner un petit histogramme. Cela est nécessaire à cause des instructions FILL de la ligne 380 car sinon la totalité de l'écran serait remplie.

La ligne 360 dessine maintenant la partie avant de l'histogramme. Viennent ensuite en ligne 370, à cause de la représentation en trois dimensions, les parties supérieure et latérale droite. L'histogramme ainsi dessiné se compose donc de trois parties séparées. Cela est rendu nécessaire par le fait que le coloriage doit intervenir en ligne 380 séparément pour chaque partie. La ligne 390 positionne alors le curseur graphique au-dessus de l'histogramme dessiné et le pourcentage arrondi (avec la "somme" comme valeur de base) est sorti. La fin de la boucle est ensuite atteinte. L'instruction TAG est annulée et le programme est terminé.

Les graphiques camembert permettent également d'obtenir d'excellents effets visuels. A partir d'un cercle, on affecte des 'parts de camembert' aux données entrées. La taille de chaque part est fonction de la grandeur relative de la valeur de donnée correspondante. On peut reprendre ici un certain nombre des

réflexions faites au sujet des histogrammes. Il faut simplement avoir présent à l'esprit qu'il n'y a qu'un seul camembert à partager, c'est-à-dire qu'on doit travailler avec de nouvelles valeurs relatives.

Enfin la réalisation du graphique se fait bien sûr de façon totalement différente des autres programmes.

```
10 REM Graphique camembert
20 CLS
30 PRINT"Combien de donnees differentes faut-il"
40 INPUT"representer (10 maximum) ";nombre
50 IF nombre>10 OR nombre<2 THEN 20
60 PRINT:PRINT"Vous pouvez entrer maintenant un"
70 PRINT"nom (3 caract. maxi) et une valeur
80 PRINT"numerique pour les donnees":PRINT
90 FOR i=1 TO nombre
100 INPUT"Nom ";nom$(i)
110 IF LEN(nom$(i))>3 THEN nom$(i)=LEFT$(nom$(i),3)
120 LOCATE 15,i+7:INPUT"Valeur ";valeur(i)
130 somme=somme+valeur(i)
140 NEXT i
150 FOR i=1 TO nombre
160 valeur360(i)=valeur(i)*360/somme+valeur360(i-1)
170 NEXT i
180 MODE 2:DEG
190 FOR i=1 TO 360
200 MOVE 320,200
210 PLOTR 150*COS(i),150*SIN(i)
220 NEXT i
230 TAG
240 FOR i=1 TO nombre
250 MOVE 320,200
260 DRAWR 150*COS(valeur360(i)),150*SIN(valeur360(i))
270 k=(valeur360(i-1)+valeur360(i))/2
280 MOVE 320+150*COS(k),200+150*SIN(k)
290 DRAWR 15*COS(k),15*SIN(k)
300 $=nom$(i)+STR$(ROUND(valeur(i)*100/somme,2))+"%"
310 IF COS(k)>0 THEN MOVER 0,8:GOTO 330
320 MOVER -8*LEN(t$),8
330 PRINT t$;
```

```
340 NEXT i
350 a$=INKEY$:IF a$="" THEN 350
360 MODE 1:END
```

Liste de variables :

a$	Chaîne renvoyant la touche enfoncée
nombre	Nombre de valeurs
i	Index de comptage
k	Variable auxiliaire
nom$(i)	Désignation des données
somme	Somme des valeurs de données
t$	Désignation de la part de camembert (Nom+pourcentage)
valeur(i)	Les valeurs de données
valeur360(i)	Les valeurs relatives

Description du programme :

10-140	Cette partie du programme est identique aux lignes 10-150 du programme d'histogrammes. On peut seulement se passer ici de la détermination de la valeur maximum. Il n'y a pas ici de nécessité absolue de limiter le nombre d'entrées à 10. Cette limite n'a été conservée que pour des raisons de lisibilité.
150-170	Calcul des valeurs relatives. Il s'agit pratiquement ici d'un calcul de pourcentage si ce n'est que la base est 360 et non 100. Comme vous le savez en effet un cercle est un angle complet de 360 degrés. On ajoute d'autre part la valeur relative précédente ("valeur360(i-1)") car les différentes parts de camembert ne seront pas ensuite dessinées relativement à la part précédente mais toujours à partir d'un point de départ.
180	Modification du mode écran et commutation sur les degrés car les valeurs des fonctions trigonométriques sont normalement sorties en radians.

190-220	Dessin d'un cercle à l'aide des fonctions trigonométriques SIN et COS. La ligne 200 sert à positionner le cercle au milieu de l'écran. Les différents points de la circonférence sont plottés (=fixés) par rapport à ce centre. Le rayon du cercle est fixé par le nombre 150 en ligne 210.
230	Activation de l'instruction TAG pour mélanger texte et graphisme.
240-340	Les parts de camembert sont dessinées dans cette boucle de programme. En ligne 350, le curseur graphique est fixé au centre de l'écran. Un point du cercle est ensuite calculé en fonction de la valeur relative "valeur360(i)" et une ligne est tracée du centre du cercle à ce point. La ligne 270 calcule la grandeur auxiliaire k. Elle sert à calculer la moitié de l'arc de cercle représenté par une part de camembert. Le curseur graphique est amené en ligne 280 sur cette coordonnée pour pouvoir ensuite dessiner en ligne 290 une courte ligne verticale vers l'arc de cercle. A la fin de cette ligne, la ligne 330 donne son nom à la part de camembert. A cet effet est formée en ligne 300 la variable t$ qui se compose du nom entré et du pourcentage représenté par la part du camembert par rapport au fromage entier. Avant la sortie de t$, le curseur graphique est encore décalé de 8 pixels vers le haut car l'angle supérieur gauche du caractère de texte est placé sur le curseur graphique. Si le milieu de l'arc de cercle d'une part de camembert se trouve placé à gauche du centre du cercle (alors COS(k)<0), le curseur graphique est encore décalé sur la gauche, sur la longueur de t$ (ligne 320). Pour pouvoir déterminer cette longueur, le pourcentage a été converti en chaîne de caractères à la ligne 300 (voyez la fonction STR$).
350-360	Attente d'une touche, modification du mode écran et fin du programme.

Comme le graphique camembert est sorti en MODE 2 (essentiellement parce que les chaînes t$ modifieraient en mode 40 colonnes toute la structure de l'écran du fait de leurs dimensions), il n'est pas possible d'utiliser des couleurs. Si vous

le souhaitez, vous pouvez bien sûr changer cela. Vous devez simplement veiller alors à ce que les caractères de texte soient représentés avec une largeur de 16 pixels. Il faut également veiller à ce que l'origine du curseur graphique ne soit pas modifiée avec ORIGIN avant le lancement du programme. Cela s'applique d'ailleurs également aux deux programmes d'histogrammes précédents.

6.8. Plotter de fonctions

Comme nous l'avons vu au chapitre précédent, les fonctions mathématiques jouent un rôle non négligeable. Nous avons utilisé les fonctions trigonométriques SIN et COS pour dessiner un cercle. Ces fonctions sont intégrées dans le BASIC, de même que LOG, EXP ou SQR. L'instruction DEF FN vous permet cependant également de définir n'importe quelles autres fonctions. Si vous avez par exemple besoin dans le cours de votre programme de la fonction simple y=19*x-7, vous pouvez faire calculer la valeur y correspondant à une valeur x quelconque en sautant chaque fois que nécessaire à un sous-programme du type :

```
1000 y=19*x-7:RETURN
```

Mais il est encore beaucoup plus simple et plus rapide d'entrer au début du programme :

```
30 DEF FN f(x)=19*x-7
```

FN f(i) vous fournira alors à la demande la valeur de fonction correspondant à une valeur quelconque de i.

De nombreux problèmes mathématiques peuvent être résolus par la représentation graphique d'une fonction ainsi définie. Surtout lorsqu'il s'agit de fonctions compliquées, il est très difficile de se faire une idée du déroulement de la fonction sans l'aide d'un ordinateur. Mais également lorsqu'il s'agit par exemple, à l'école, de commenter la forme d'une courbe, un plotter de fonction fournit un excellent moyen de contrôle.

Voici maintenant un programme qui vous permet d'inscrire une fonction en ligne 30. Cette fonction sera ensuite dessinée après que vous ayez entré une zone x. Nous avons choisi ici comme exemple la fonction SIN(x) qui est assez connue. Soulignons encore une fois que vous devez modifier la ligne 30 pour faire dessiner une autre fonction de votre choix.

```
10 REM Plotter de fonctions
20 CLS
30 DEF FN f(x)=SIN(x)
40 PRINT"Vous pouvez examiner n'importe quelles"
50 PRINT"fonctions en modifiant la ligne 30"
60 PRINT"du programme. ":PRINT:PRINT:PRINT
70 PRINT"Entrez les valeurs maxi et mini de X"
80 PRINT"qu'il faut prendre en compte:":PRINT:PRINT
90 INPUT"Valeur mini de X ";kw:PRINT
100 INPUT"Valeur maxi de X ";gw
110 IF gw<=kw THEN CLS:GOTO 40
120 sw=(gw-kw)/640
130 REM Dessin des axes de coordonnees
140 MODE 2
150 PLOT 0,400:DRAW 0,0:PLOT 0,200:DRAW 640,200
160 LOCATE 2,14:PRINT kw
170 gw$=STR$(gw):LOCATE 80-LEN(gw$),14:PRINT gw
180 REM Determination des valeurs maxi et mini de la fonction
190 mif=FN f(kw):maf=mif
200 FOR i=kw TO gw STEP sw*10
210 IF FN f(i)<mif THEN mif=FN f(i)
220 IF FN f(i)>maf THEN maf=FN f(i)
230 NEXT i
240 REM Les valeurs de fonction calculees sont relativisees pour
la sortie ecran
250 IF ABS(maf)>ABS(mif) THEN rf=200/maf:LOCATE 2,1:PRINT
ROUND(maf,2):GOTO 280
260 rf=-200/mif:LOCATE 2,25:PRINT ROUND(mif,2)
270 REM Dessin de la courbe
280 FOR i=kw TO gw STEP sw
290 j=j+1:PLOT j,FN f(i)*rf+199
300 NEXT i
310 x$=INKEY$:IF x$="" THEN 310
320 MODE 1:END
```

Liste de variables :

gw	Valeur maxi de x
gw$	gw sous forme de chaîne
i	Index de comptage
j	Coordonnée x (domaine graphique) pour dessiner la courbe
kw	Valeur mini de x
maf	Valeur de fonction maximale
mif	Valeur de fonction minimale
rf	Facteur relatif (rapporté à la taille de l'écran)
sw	Pas
x$	Chaîne renvoyant la valeur de la touche enfoncée

Description du programme :

10-20	Commentaire et vidage de l'écran
30	Définition de la fonction.
40-60	Sortie d'un commentaire sur la ligne 30
70-110	Entrée des valeurs mini et maxi de x. La ligne 110 vérifie que la valeur maxi est bien supérieure à la valeur mini.
120	Comme jusqu'à 640 valeurs de fonction différentes peuvent être plottées pour la courbe à dessiner, il est nécessaire de diviser l'intervalle x par 640. Cette valeur sera affectée à la variable sw.
130-170	Après une modification du mode écran, les axes de coordonnées sont dessinés en ligne 150 et on écrit sur l'axe des x (lignes 160-170). La ligne 170 s'assure à cet égard que la valeur maxi de x soit sortie à l'extrémité droite de l'axe des x, indépendamment de sa 'longueur'.

180-230	Les valeurs mini et maxi de la fonction sont calculées. Les variables correspondantes mif et maf se voient tout d'abord affecter une valeur initiale en ligne 190. Cette valeur initiale peut être d'une manière générale n'importe quelle valeur de fonction dans l'intervalle x indiqué. On compare ensuite dans une boucle (lignes 200-230) chaque valeur de fonction à mif ou à maf. Chaque fois qu'une valeur de fonction est inférieure à mif ou supérieure à maf, mif ou maf se voient attribuer cette valeur de fonction. Le pas de cette boucle est sw*10. Il est également possible de n'employer que sw comme pas. Vous obtiendrez alors des résultats plus précis pour mif et maf mais le temps de calcul sera nettement plus long.
240-260	Comme on ne dispose que de 400 pixels dans le sens de la verticale, les valeurs de fonction pour la sortie écran doivent être relativisées. On examine pour cela quelle valeur absolue (c'est-à-dire en faisant abstraction du signe) est la plus élevée, de celle de mif ou de maf. La valeur la plus élevée est alors utilisée pour le calcul du facteur rf. Cette valeur absolue est également utilisée pour marquer l'axe des y.
270-300	La courbe est plottée dans la boucle de programme (lignes 280-300). En raison du calcul de sw en ligne 120, j passe ici de 1 à 640. Le facteur rf fait en sorte que la valeur absolue la plus élevée soit plottée sur le bord supérieur ou inférieur de l'écran. La totalité de l'écran est donc remplie indépendamment de la dimension des valeurs de fonction.
310-320	'Attente', modification du mode écran et fin du programme.

Si la fonction entrée fait apparaître de grandes différences dans la valeur de fonction même pour de faibles modifications de la valeur x, alors il est préférable sur le plan visuel de travailler avec l'instruction DRAW plutôt qu'avec PLOT. La ligne 290 du programme se présenterait donc ainsi :

```
290 j=j+1:DRAW j,FN f(i)*rf+199
```

Il faudrait encore amener le curseur graphique dans une position de départ :

```
280 PLOT 0,FN f(kw)*rf+199
285 FOR i=kw TO gw STEP sw
```

Du fait de l'instruction de saut de la ligne 250, on ne peut pas choisir le numéro de ligne 275 pour l'instruction PLOT supplémentaire.

Après cette modification du programme, vous pourrez obtenir avec la fonction SIN de très beaux effets graphiques si vous choisissez un intervalle x très large. Essayez par exemple de faire tourner le programme avec un intervalle x de -500 à 500...

Si vous voulez disposer d'informations plus précises sur le déroulement de la fonction, vous pouvez par exemple affecter en lignes 210 et 220 les valeurs x des points extrêmes (mif et maf) à deux variables supplémentaires et faire également sortir ces informations :

```
210 IF FN f(i)<mif THEN mif=FN f(i):xmif=i
220 IF FN f(i)>maf THEN maf=FN f(i):xmaf=i
314 LOCATE 1,3
315 PRINT"Maximum pour x =";xmaf:PRINT
316 PRINT"Minimum pour x =";xmif
317 x$=INKEY$:IF x$="" THEN 317
```

Bien entendu il faudrait aussi que vous travailliez alors en ligne 200 avec le pas sw et non sw*10 pour obtenir des résultats précis.

Lorsqu'on examine le déroulement d'une fonction, il est également intéressant de connaître le(s) point(s) où la fonction croise l'axe des x (point zéro). Si vous voulez avoir des indications plus précises à ce sujet (le dessin vous donne déjà des informations approximatives à cet égard), vous devez ajouter des instructions supplémentaires dans la partie du programme composée des lignes 180 à 230. Nous vous en laissons le soin à titre d'exercice. Un truc: la fonction coupe l'axe

des x au moment exact où le signe des valeurs de fonction change.

6.9. Graphiques en trois dimensions

L'écran est comme une surface plane en deux dimensions (longueur et largeur). Lorsque vous voulez représenter sur cette surface des corps spatiaux, vous devez trouver un moyen de simuler la troisième dimension sur le plan visuel. Le présent chapitre vous fait trois propositions dans ce sens.

Nous avions déjà vu dans le dernier chapitre comment la fonction sinus permet d'obtenir des effets graphiques intéressants. Nous allons utiliser ici la fonction sinus pour simuler la troisième dimension. Le principe est très simple: on dessine une fonction sinus mais pas avec des points mais avec des lignes joignant un point (0,200) au point actuel de la courbe sinus. L'illusion d'optique est alors provoquée par le chevauchement des lignes.

Le programme correspondant repose sur le plotter de fonctions du dernier chapitre. Il vous permet donc d'examiner également les effets de chevauchement produits par d'autres fonctions.

Et voici comment se présente le programme 'Graphisme avec fonctions' :

```
10 REM Graphisme avec fonctions
20 CLS
30 DEF FN f(x)=SIN(x)
40 PRINT"Vous pouvez examiner n'importe quelles"
50 PRINT"fonctions en modifiant la ligne 30"
60 PRINT"du programme. ":PRINT:PRINT:PRINT
70 PRINT"Entrez les valeurs maxi et mini de X"
80 PRINT"qu'il faut prendre en compte:":PRINT:PRINT
90 INPUT"Valeur mini de X ";kw:PRINT
100 INPUT"Valeur maxi de X ";gw:PRINT:PRINT:PRINT
110 IF gw<=kw THEN CLS:GOTO 40
120 sw=(gw-kw)/640
130 PRINT"Veuillez entrer une valeur pour la"
```

```
140 PRINT"densite des lignes du graphisme"
150 PRINT"(valeur pos., plus elle est faible"
160 INPUT"plus les lignes seront serrées) ";z
170 MODE 2
180 ORIGIN 0,200
190 mif=FN f(kw):maf=mif
200 FOR i=kw TO gw STEP sw*10
210 IF FN f(i)<mif THEN mif=FN f(i)
220 IF FN f(i)>maf THEN maf=FN f(i)
230 NEXT i
240 IF ABS(maf)>ABS(mif) THEN rf=200/maf ELSE rf=-200/mif
250 EVERY z GOSUB 310
260 FOR i=kw TO gw STEP sw
270 j=j+1:MOVE j,FN f(i)*rf-1
280 NEXT i
290 x$=INKEY$:IF x$="" THEN 290
300 MODE 1:END
310 DRAW 0,0:RETURN
```

Comme vous le voyez, ce programme n'a pas besoin d'axes de coordonnées. D'autre part, l'origine du curseur graphique est fixée sur le point 0,200 (ligne 180) et les points de la courbe ne sont pas plottés (ligne 270).

Ce qui est nouveau dans ce programme, ce sont les lignes 130-160 et 310.

En fonction d'une variable de délai z, qui est fixée dans les lignes 130-160, un appel de sous-programme est effectué en ligne 250 (ligne 310). Dans ce sous-programme, une ligne est tracée de la position actuelle du curseur graphique (ligne 270) à l'origine du graphique. Rappelons que cette opération dépend uniquement du temps écoulé et qu'elle est indépendante du déroulement du programme principal (voyez aussi à ce sujet le chapitre 3.6).

Le mieux est que vous essayiez immédiatement le programme. Vous pourriez commencer avec un intervalle de 0 à 6 et une 'densité' de 10. Vous reconnaîtrez alors une sorte d'éventail. Vous pourrez ensuite essayer aussi de modifier l'instruction

DRAW de la ligne 310 ou la fonction de la ligne 30. De nombreux effets très intéressants sont ainsi rendus possibles.

Une autre possibilité de représenter des graphiques en trois dimensions consiste à travailler avec divers types de hâchures. Le programme suivant réalise un dessin dans lequel vous pouvez distinguer de nombreuses images, une toile d'araignée ou une pyramide vue d'en haut ou encore bien d'autres choses...

```
10 REM Graphisme simulant les trois dimensions
20 INPUT"1er pas ";s1
30 INPUT"2nd pas ";s2
40 CLS
50 ORIGIN 320,200
60 FOR i=0 TO 190 STEP s1
70 PLOT i,j:DRAW i,-j
80 PLOT -i,j:DRAW -i,-j
90 PLOT i,j:DRAW -i,j
100 PLOT -i,-j:DRAW i,-j
110 j=j+s2
120 NEXT i
130 x$=INKEY$:IF x$="" THEN 130
140 CLS:END
```

La densité des traits parallèles dépend des pas choisis. Essayez vous-même de reconstituer la taille des paramètres PLOT et DRAW. Cette fois, ce n'est vraiment pas difficile. Mais faites des essais avec le programme. Vous pouvez par exemple transformer la ligne 60 en :

```
60 FOR i=190 TO 0 STEP -1
```

Vous obtiendrez alors une image toute différente.

Un troisième graphique vous présentera maintenant un parallélépipède tel que ceux que vous avez étudiés en géométrie. Mais ce parallélépipède ne sera pas dessiné qu'une fois mais plusieurs fois de différents côtés. Alors que le côté avant ou arrière du parallélépipède ne bouge pas, les autres côtés se modifient sans cesse. Le mieux est que vous regardiez vous-même le résultat produit par le programme (l'origine

graphique doit être placée sur le point 0,0 avant le lancement du programme).

```
10 REM Parallelepipede
20 MODE 2
30 s1=98:s2=-98:sw=-4:fk=1
40 FOR a=s1 TO s2 STEP sw
50 PLOT 200,100,1
60 DRAW 200,300,1:DRAW 400,300,1:DRAW 400,100,1:DRAW 200,100,1
70 b=(98-ABS(a))*fk
80 PLOT 200,300,f:DRAWR b,a,f:DRAWR 200,0,f
90 PLOT 400,300,f:DRAWR b,a,f:DRAWR 0,-200,f
100 PLOT 400,100,f:DRAWR b,a,f:DRAWR -200,0
110 PLOT 200,100,f:DRAWR b,a,f:DRAWR 0,200
120 FOR i=1 TO 200:NEXT i
130 IF f=1 THEN f=0:GOTO 80
140 f=1:NEXT a
150 s1=-s1:s2=-s2:sw=-sw:fk=-fk:GOTO 40
```

Peut-être cette image vous rappelle-t-elle un dessin technique ou peut-être aurez-vous simplement une impression de vertige en regardant cette image. En tout cas, il y a, ici aussi, un effet de trois dimensions.

Vous ne devriez pas avoir de peine à comprendre les différentes instructions utilisées dans le programme. Voici malgré tout quelques explications :

Les côtés fixes de devant et de derrière sont chaque fois réalisés par les lignes 50-60 alors que la troisième dimension est dessinée dans les lignes 80-110. L'indication du crayon de couleur f, en lignes 80-110, sert à effacer la troisième dimension déjà dessinée pour parvenir à une représentation continue (voir la ligne 130). La vitesse de la représentation est déterminée par la valeur finale de la boucle d'attente de la ligne 120. Vous pouvez naturellement modifier à votre guise cette valeur finale. Les paramètres a et b des instructions PLOT et DRAW donnent toujours une somme absolue de 98 (voir la ligne 70) pour assurer une représentation à angle droit (voyez aussi les valeurs affectées aux variables s1 et s2 à la ligne 30). La variable fk a pour fonction de dessiner également la troisième dimension à la

gauche du côté fixe (après 180 degrés). De même que les valeurs initiale, finale et de pas de la boucle de programme des lignes 40 à 140, la variable fk change de signe après chaque parcours de la boucle (voir la ligne 150). Chaque parcours de la boucle concerne par conséquent soit le côté droit, soit le côté gauche. Vous pouvez bien sûr tout à fait modifier aussi le pas de la boucle de programme, sw. Nous vous proposons donc de vous livrer vous-même à telle ou telle de ces modifications du programme.

6.10. Le Joystick et les jeux

Les jeux électroniques tirent leur intérêt d'un graphisme efficace. Comme le graphisme de votre CPC 6128 est vraiment de grande qualité, vous pouvez programmer de très beaux jeux par des moyens relativement simples (nous pensons ici notamment aux orgies de POKEs nécessaires sur d'autres ordinateurs).

Dans de nombreux jeux électroniques le succès du jeu dépend de la rapidité de réaction du joueur. Le clavier de votre ordinateur est alors vite dépassé ou se révèle pour le moins peu pratique. Un joystick (manche à balai) permet par contre une réaction beaucoup plus rapide. Le joystick est un manche de commande qui se compose d'une poignée et d'un bouton de 'feu'.

Alors que la poignée est conçue pour diriger un objet (par exemple une balle ou un petit bonhomme ou un obstacle, etc...) le bouton 'feu' sert le plus souvent à 'tirer'.

Vous pouvez connecter jusqu'à deux joysticks à votre CPC. Les différentes fonctions du joystick peuvent être interrogées avec INKEY ou INKEY$ (les joysticks sont donc traités comme faisant partie du clavier) ou en utilisant les fonctions JOY(0) et JOY(1). L'argument 0 de cette fonction correspond bien sûr ici au premier joystick, l'argument 1 correspondant au second.

Pour ne pas compliquer les choses inutilement, nous ne nous intéresserons dans les explications suivantes qu'au cas où l'on

emploie un seul joystick. Toutefois ces développements valent aussi pour le second joystick.

Pour bien comprendre le fonctionnement du joystick, entrez donc le petit programme suivant après avoir connecté un joystick à votre ordinateur :

```
10 PRINT JOY(0)
20 GOTO 10
```

Si vous actionnez maintenant votre joystick ou si vous appuyez sur le bouton 'feu' vous pourrez observer chaque fois une modification du nombre renvoyé par le programme. Si vous déplacez par exemple la poignée vers le haut, le chiffre 1 sera sorti.

Le nombre renvoyé par la fonction JOY est codé bit par bit. Ce n'est pas ici le lieu d'expliquer en détail les notions de bits et d'octets (voyez à ce sujet le chapitre 4.5). Indiquons simplement qu'à chaque action possible avec le joystick correspond un bit ayant la signification suivante :

Bit	Valeur décimale	Signification
0	1	vers le haut
1	2	vers le bas
2	4	gauche
3	8	droite
4	16	feu 2
5	32	feu 1

Si vous tirez par exemple la poignée du joystick en transversale vers en haut à gauche, alors JOY vous renverra la valeur 5. Vous pouvez donc également identifier les combinaisons de fonctions du joystick par simple addition des valeurs

correspondantes. Les différentes valeurs sont chaque fois des puissances de 2. La valeur 1 est par exemple égale à 2^0, la valeur 2 à 2^1, etc... Si une valeur décimale contient une puissance de 2, cela veut dire que le bit correspondant est mis. Le nombre décimal 10 contient par exemple les puissances 2^3 et 2^1, cela veut donc dire que le premier (vers le bas) et le troisième (vers la droite) bits sont mis. Le nombre binaire correspondant sera : 001010 (les deux zéros du début servent à marquer qu'on utilise bien 6 bits en tout).

La représentation en format binaire ne présente bien sûr que des avantages si vous voulez optimiser la programmation du joystick. Mais nous y reviendrons à la fin de ce chapitre.

Nous allons d'abord vous présenter un programme d'exemple. Une fois ce programme lancé, vous pourrez déplacer un petit bonhomme sur l'écran avec le joystick. Pour représenter ce petit bonhomme, nous utilisons le code CHR$(249) (ou CHR$(251) quand le bonhomme va à gauche et CHR$(250) quand il va à droite). Et voici comment le programme se présente :

```
10 REM Bonhomme joystick
20 x1=39:y1=12
30 MODE 2
40 IF JOY(0)-32=1 AND y1>1 THEN LOCATE x1,y1:PRINT"  ":y1=y1-
1:LOCATE x1,y1:FRAME:PRINT CHR$(249)
50 IF JOY(0)-32=2 AND y1<25 THEN LOCATE x1,y1:PRINT"  ":y1=y1+1:
LOCATE x1,y1:FRAME:PRINT CHR$(249)
60 IF JOY(0)-32=4 AND x1>2 THEN LOCATE x1,y1:PRINT"  ":x1=x1-
1:LOCATE x1,y1:FRAME:PRINT CHR$(250)
70 IF JOY(0)-32=8 AND x1<78 THEN LOCATE x1,y1:PRINT"  ":x1=x1+1:
LOCATE x1,y1:FRAME:PRINT CHR$(251)
80 IF JOY(0)-32=5 AND y1>1 AND x1>2 THEN LOCATE x1,y1:PRINT"  ":
x1=x1-1:y1=y1-1:LOCATE x1,y1:FRAME: PRINT CHR$(250)
90 IF JOY(0)-32=9 AND y1>1 AND x1<78 THEN LOCATE x1,y1:PRINT"  ":
x1=x1+1:y1=y1-1:LOCATE x1,y1:FRAME: PRINT CHR$(251)
100 IF JOY(0)-32=6 AND y1<25 AND x1>2 THEN LOCATE x1,y1:PRINT"  ":
x1=x1-1:y1=y1+1:LOCATE x1,y1:FRAME: PRINT CHR$(250)
110 IF JOY(0)-32=10 AND y1<25 AND x1<78 THEN LOCATE x1,y1:PRINT"
": x1=x1+1:y1=y1+1:LOCATE x1,y1:FRAME: PRINT CHR$(251)
120 GOTO 40
```

Après que des valeurs initiales aient été affectées aux coordonnées x1 et y1 (ligne 20) et qu'on soit passé en mode 80 colonnes (ligne 30) le joystick est interrogé dans les lignes 40 à 110. Cette interrogation se déroule fondamentalement comme celle du clavier avec INKEY et INKEY$. En raison de l'instruction de saut de la ligne 120 cet examen de l'état du joystick et le déplacement du bonhomme qui en est la conséquence se poursuivent jusqu'à ce que vous appuyiez deux fois sur la touche ESC.

Toute interrogation du joystick est reliée à une vérification des coordonnées. Si le petit bonhomme est par exemple déplacé vers la gauche avec JOY(0)=4, encore faut-il que sa position ne soit pas zéro car cela entraînerait un message d'erreur. La vérification des coordonnées dépend donc chaque fois de la direction du déplacement. Pour le reste, le principe de ce programme est très simple. Après qu'une des valeurs énumérées ait été reçue pour JOY dans les lignes 40 à 110 et à condition que le bonhomme ne menace pas de quitter l'écran (vérification des coordonnées), le bonhomme actuellement sur l'écran est effacé à l'aide d'une chaîne vide. Les coordonnées sont ensuite à nouveau déterminées et le bonhomme apparaît exactement dans cet emplacement. L'interrogation se poursuit ensuite. La rapidité du déroulement du programme crée l'illusion du mouvement. Ce principe a d'ailleurs déjà été présenté au chapitre 6.4.

On peut distinguer les directions suivantes :

Ligne	40:	vers le haut
Ligne	50:	vers le bas
Ligne	60:	vers la gauche
Ligne	70:	vers la droite
Ligne	80:	transversalement vers en haut à gauche
Ligne	90:	transversalement vers en haut à droite

Ligne	100:	transversalement vers en bas à gauche
Ligne	110:	transversalement vers en bas à droite

Si vous lancez le programme, vous constaterez que le mouvement horizontal est relativement lent. C'est le mode 80 colonnes qui en est responsable. Alors qu'il n'y a que 25 lignes à parcourir verticalement pour passer d'un bord à l'autre de l'écran, il y en a tout de même 80 dans le sens horizontal. Si vous voulez changer cela, vous pouvez passer au mode 40 colonnes (il faudra alors modifier également les limites des coordonnées) ou bien vous pouvez accélérer le mouvement horizontal en sautant quelques cases, par exemple en écrivant x1=x1+3 au lieu de x1=x1+1. Mais dans ce cas également il conviendra de changer les limites des coordonnées dans les lignes de programme correspondantes.

Le programme montre bien à quelle vitesse le joystick permet de déplacer un objet sur l'écran. Mais c'est tout, autrement dit : le programme est ennuyeux. A vous de compléter le programme pour donner un sens à ces mouvements en développant un jeu intéressant. Vous pourriez par exemple faire entrer en jeu la 'bombe' que nous avions vue au chapitre 6.4. Et si les coordonnées x2 et y2 de la bombe sont identiques aux coordonnées x1 et y1 du bonhomme, alors... ça craint !

Mais pour le moment nous nous contenterons de disperser quelques morceaux de roche sur l'écran, morceau de roche que votre bonhomme devra essayer de "manger". Cela peut aisément être programmé grâce à quatre lignes de programme supplémentaires :

```
33 FOR i=1 TO 30
34 LOCATE INT(RND*77+2),INT(RND*24+1)
35 PRINT CHR$(143)
36 NEXT i
```

Vous voyez que le petit bonhomme peut être déplacé tellement vite qu'une commande précise devient souvent difficile. Il vous

faut pour le moins du doigté et cette extension du programme est un excellent moyen pour en acquérir.

L'aspect de compétition qui manque ici peut par ailleurs être créé grâce à d'autres extensions du programme.

Vous pourriez par exemple intégrer un chronomètre (fonction TIME) ou commander avec un second joystick des morceaux de roche (CHR$(127) par exemple) ou un second bonhomme. Il pourrait s'agir par exemple de "manger" les morceaux de roche. En tout cas laissez libre cours à votre imagination.

Dans cet exemple de programme, les boutons 'feu' ne jouaient aucun rôle. Si vous voulez leur attribuer une fonction et si vous voulez combiner cette fonction avec les huit directions possibles, cela entraînerait une extension non négligeable du programme. Vous devriez alors travailler avec une autre technique de programmation. Il vous faudrait pour cela convertir en un nombre binaire de 6 chiffres les valeurs décimales renvoyées par JOY. Vous pourriez ensuite tester les 6 chiffres de droite à gauche pour voir si le bit correspondant doit être fixé (valeur 1) ou non (valeur 0). On sauterait ensuite à différents sous-programmes. Chaque sous-programmes correspondrait donc à une fonction du joystick. Le nombre binaire 001001 entraînerait par exemple un saut au sous-programme "droite" puis au sous-programme "bas".

Nous en resterons cependant à ces explications théoriques en ce qui concerne cette technique de programmation. Nous vous conseillons plutôt d'utiliser cette manière de procéder lorsque vous serez confronté à des programmes avec joystick plus complexes. Mais pour illustrer encore une fois ce principe par un petit programme, voici une proposition pour l'identification des différentes valeurs de bits :

```
10 INPUT"Nombre entre 1 et 63 ";z
20 a$=BIN$(z,6)
30 FOR i=1 TO 6
40 b$=RIGHT$(a$,i)
50 b$=LEFT$(b$,1)
```

```
60 PRINT b$
70 NEXT i
```

Si vous entrez par exemple la valeur 10, vous obtiendrez le résultat suivant :

```
0   (Bit 0)
1   (Bit 1)
0   (Bit 2)
1   (Bit 3)
0   (Bit 4)
0   (Bit 5)
```

La variable b$ renvoie donc les chiffres du nombre binaire de droite à gauche. La limite du nombre décimal à entrer est calculée d'après la formule suivante : 20+21+22+23+24+25 = 63. JOY fournit donc la valeur 63, lorsque toutes les fonctions du joystick sont combinées les unes avec les autres. Cela n'est d'ailleurs pas possible en réalité car chaque fonction exclut la plupart des autres. C'est ainsi que les fonctions "haut" et "bas" ne peuvent évidemment pas être combinées.

Si vous n'êtes pas encore très à l'aise dans le maniement des nombres binaires, alors étudiez un peu le programme "Aide à la détermination de la valeur de canal" du chapitre 6.3. Nous ne travaillons pas dans ce programme avec la fonction BIN$ mais nous recherchons par un raisonnement mathématique quelles puissances de 2 apparaissent dans une valeur décimale.

6.11. L'utilisation des 128 K

La mémoire du CPC a 128 K de RAM répartis en deux unités de 64 K chacune. Alors que CP/M Plus travaille toujours avec la totalité des 128 K de place mémoire, le BASIC n'utilise en général que les 64 premiers K. Vous aimeriez certainement bien savoir comment le second bloc de 64 K peut être utilisé pour la programmation en BASIC.

Par rapport à d'autres ordinateurs 128 K, ce problème a très bien été résolu sur le CPC 6128. Sur la première face de votre

jeu de disquettes système figure en effet un programme appelé "BANK MANAGER". Si vous lancez ce programme avec :

```
RUN"BANKMAN",
```

vous disposerez quelques secondes plus tard des instructions RSX suivantes (RSX = Resident System eXtension) :

```
SCREENCOPY
SCREENSWAP
BANKOPEN
BANKWRITE
BANKREAD
BANKFIND
```

Ces instructions supplémentaires sont marquées par un trait vertical (touche numéro 26 avec SHIFT) les précédant. Après que le programme BANKMAN ait été chargé, ces instructions peuvent être utilisées dans des programmes BASIC pour employer le second bloc de 64 K. Alors que les deux premières instructions permettent d'amener et d'échanger des pages écran sur l'écran, les autres instructions servent à stocker des blocs de texte (chaînes de caractères). Nous n'étudierons dans ce chapitre que la première possibilité. L'emploi des autres instructions est expliqué en détail au chapitre 8.3. Vous trouverez également dans ce chapitre un exemple d'application intéressant puisqu'il s'agit de la programmation de fichiers relatifs avec le second bloc de 64 K employé comme disque RAM.

Pour stocker le contenu d'une page écran, votre CPC a besoin de 16384 octets, c'est-à-dire exactement 16 K. L'ordinateur a besoin de ce volume de mémoire que l'écran soit plein ou qu'il soit presque vide. Après le lancement de "BANKMAN" vous disposez de cinq pages écran logiques de 16 K chacune. Le numéro correspondra toujours à la page visible. Les quatre autres pages (numéros 2 à 5) se trouvent dans la RAM supplémentaire. Vous pouvez donc maintenant copier ou échanger des images. Avec :

```
|SCREENCOPY,2,1
```

vous pourrez par exemple copier l'image visible sur l'écran dans la seconde page. Vous pourrez alors vider l'écran et produire une nouvelle image que vous pourrez alors copier dans la troisième page avec :

```
|SCREENCOPY,3,1
```

Avec cette instruction le premier paramètre indique toujours dans quelle page doit être placée la copie alors que le second paramètre détermine quelle page doit être copiée. La syntaxe de cette instruction est donc aussi simple que possible.

Il est tout aussi simple d'échanger des images entre elles. Avec :

```
|SCREENSWAP,2,3
```

vous échangerez par exemple le contenu des deux pages indiquées. Il ne se passera toutefois rien sur l'écran puisque seule la première page est visible. Mais si vous entrez maintenant :

```
|SCREENSWAP,1,2
```

vous verrez sur l'écran l'image que nous avions copié plus haut dans la troisième page.

Un programme simple va nous permettre d'illustrer le maniement de ces instructions :

```
10 MODE 1
20 PRINT TAB(16)"Image 1"
30 FOR i=1 TO 960:PRINT"1";:NEXT
40 |SCREENCOPY,2,1
50 CLS
60 PRINT TAB(16)"Image 2"
70 FOR i=1 TO 960:PRINT"2";:NEXT
80 |SCREENCOPY,3,1
90 |SCREENCOPY,1,2
100 |SCREENSWAP,3,2
110 GOTO 90
```

Les lignes 20 et 30 produisent ici une image qui est alors copiée dans la seconde page (ligne 40). L'écran est ensuite vidé et une nouvelle image est réalisée (lignes 60 et 70) qui est alors copiée dans la troisième page (ligne 80). Dans le déroulement ultérieur du programme (voir la ligne 110) on amènera toujours sur l'écran l'image qui se trouve dans la seconde page (ligne 90).

Du fait de l'instruction d'échange de la ligne 100 les deux images produites plus haut seront alors échangées en permanence.

Les deux instructions SCREEN permettent d'obtenir, notamment dans des programmes assez complexes, de très jolis effets. On peut par exemple placer dans les pages 2 à 5 des écrans d'aide avec des explications détaillées sur le mode d'emploi du programme ou encore des graphiques supplémentaires (par exemple des histogrammes ou des graphiques camembert). Vous trouverez au chapitre suivant un programme qui utilise le second bloc de 64 K de cette manière. Ce n'est pas un hasard s'il s'agit d'un programme graphique. Vous pouvez naturellement utiliser les instructions SCREEN également pour des textes écran. Vous devez cependant veiller dans ce cas à ce que l'écran ne défile pas vers le haut faisant ainsi disparaître une partie du texte. Il ne faut pas non plus que les modes écran des images diffèrent. Or cela peut arriver assez souvent pour les dessins.

6.12. Editeur graphique

Il vous est certainement déjà arrivé de regretter qu'il soit relativement difficile de concrétiser sur l'écran des idées de graphisme en utilisant les différentes instructions graphiques. En effet, lorsque vous utilisez le support traditionnel de la pensée qu'est le papier, vous pouvez réaliser assez facilement de petites modifications alors que le graphisme sur ordinateur nécessite un calcul précis des coordonnées avant que l'instruction graphique correspondante ne puisse être entrée.

Un éditeur graphique peut vous faciliter considérablement le travail dans ce domaine. Une fois le programme lancé vous

pouvez en effet déplacer le curseur graphique, dessiner des lignes ou des cercles, modifier les couleurs, etc..., en actionnant simplement quelques touches. Comme il n'est cependant pas possible que le clavier de l'ordinateur soit d'emploi aussi commode que ce 'bon vieux' crayon, de nouvelles interfaces entre l'homme et l'ordinateur ont été développée ces dernières années. Outre le joystick, que nous vous avons déjà présenté, citons notamment les instruments appelés "souris" et "tableur graphique".

Le tableur graphique est d'ailleurs très proche du papier de par son mode d'emploi. Nous ne prendrons cependant pas en compte ces appareils d'entrée dans les développements qui vont suivre. Un exposé détaillé du traitement graphique professionnel dépasserait en effet largement le cadre de cet ouvrage. Si vous disposez toutefois de l'instrument relativement simple qu'est le joystick, alors il vous est possible de l'utiliser dans le programme suivant après avoir apporté à ce dernier les quelques modifications nécessaires (voyez le chapitre 6.10).

Le programme suivant vous permet de réaliser n'importe quelles images que vous pourrez ensuite sauvegarder sur disquette pour les recharger à la demande. Comme le programme utilise l'instruction RSX SCREENCOPY, vous devez faire tourner le programme BANKMAN avant de lancer le programme. Cette instruction vous permet de gérer quatre images différentes simultanément. Il vous suffit de frapper une touche pour amener une autre image sur l'écran. Vous pouvez alors travailler sur cette nouvelle image. Les 64 de RAM supplémentaire sont donc utilisés au mieux. Il vous suffit également de frapper une touche pour faire afficher l'affectation des touches du clavier. En frappant à nouveau sur une touche vous pouvez alors ramener sur l'écran l'image précédente. C'est également de cette manière que sont entrés les paramètres nécessaires pour dessiner un cercle ou pour revenir au menu de départ.

Le menu se compose des parties suivantes :

- Créer ou afficher image
- Charger image
- Stocker image
- Sortie de l'affectation des touches
- Contenu de la disquette
- Fin du programme

Si vous choisissez l'une des trois premières fonctions, on vous demande ensuite d'indiquer un numéro d'image ou un numéro de départ. S'il s'agit du chargement ou de la sauvegarde, l'image est copiée avec SCREENCOPY dans la page indiquée (voir le chapitre 6.11) ou bien est transférée de la page indiquée sur la disquette. Il faut pour cela indiquer chaque fois un nom d'image. Pour créer une image, vous n'indiquez par contre qu'un numéro de départ, c'est-à-dire que l'image sera copiée plus tard dans la page portant le numéro indiqué. Pendant que vous êtes en train de réaliser un dessin, ce numéro apparaît dans l'angle supérieur gauche de l'écran. Nous avons intégré dans le programme la fonction "contenu disquette" car vous devez absolument indiquer un nom correct pour le chargement d'une image. En effet, un nom incorrect entraînerait un glissement interdit de l'écran. Vous pouvez donc vous informer ainsi sur les images existant sur la disquette avant de charger une image. La fonction "Sortie des affectations de touches" peut être appelée même pendant que vous êtes en train de réaliser une image. Elle peut cependant également être appelée à partir du menu. Vous pouvez donc disposer à tout moment des explications nécessaires si vous n'avez pas sous la main les explications que nous allons maintenant vous donner.

Mais venons-en à l'affectation des touches du clavier :

Les touches de commande du curseur vous permettent de modifier les coordonnées écran qui sont affichées en haut à droite (coordonnée y) et en bas à droite (coordonnée x) de l'écran. Une modification de coordonnées n'entraîne pas cependant de déplacement du curseur graphique. Ce n'est que

lorsque vous actionnez la touche TAB que le curseur graphique se déplace avec MOVE vers la position indiquée. Si vous voulez dessiner une ligne, utilisez la touche COPY. Celle-ci vous permet de tirer une ligne de l'emplacement du curseur graphique à la position indiquée en haut à droite et en bas à droite. Vous pouvez donc appuyer simultanément sur les touches de commande du curseur et sur la touche COPY pour dessiner effectivement. Vous n'utiliserez en général la touche TAB que lorsque vous voudrez réaliser une image disjointe. Si vous trouvez que la modification des coordonnées s'effectue trop lentement, vous pouvez l'accélérer avec la touche plus ou la ralentir à nouveau avec la touche moins.

Si vous actionnez la touche "k" ou "g" l'image sera copiée dans la RAM supplémentaire et vous pourrez entrer tranquillement les paramètres "Rayon" et "Angle" (360 degrés=cercle complet) permettant de créer un cercle.

L'image actuelle apparaîtra alors ensuite à nouveau sur l'écran, c'est-à-dire que l'image copiée dans la RAM supplémentaire reviendra dans la première page. La différence entre "k" et "g" est que "k" dessine un simple cercle alors que "g" dessine un cercle plein. Après appel de la fonction cercle, c'est la position indiquée en haut à droite et en bas à droite de l'écran qui sera prise comme centre du cercle.

Les touches "w", "s", "r" et "b" vous permettent de déterminer la couleur de votre 'crayon'. Comme le programme tourne en mode 1, vous pouvez choisir parmi quatre couleurs. "w" signifie blanc, "s" noir, "r" rouge et "b" bleu. Si vous choisissez la couleur blanche vous pouvez supprimer les points et lignes existants comme avec une gomme puisque la couleur du fond est également le blanc.

Si vous appuyez sur la touche espace ou ENTER l'image actuelle sera également copiée dans la RAM supplémentaire. Alors que la touche espace fait sortir l'affectation des touches et qu'en appuyant à nouveau sur cette touche vous faites réapparaître l'image actuelle sur l'écran, la touche ENTER vous permet de revenir au menu.

La dernière possibilité intéressante offerte par le programme consiste à faire échanger l'image actuelle (page 1) avec une autre image placée dans la RAM supplémentaire. L'image actuelle sera alors copiée dans la page correspondante (voir la variable n) et l'image voulue apparaîtra sur l'écran. Pour effectuer ce choix d'une image, il vous suffit de choisir simplement le nombre 2, 3, 4 ou 5 parmi les touches numériques placées en haut à gauche de votre clavier. Cette puissante fonction vous permet d'amener par exemple rapidement sur l'écran des dessins de détail. Si vous avez par exemple réalisé une image représentant l'esquisse d'une maison, vous pouvez alors appeler les dessins des différentes pièces en frappant simplement une touche.

Après toutes ces explications sur le mode d'emploi du programme, vous êtes certainement impatient d'en découvrir le listing. Le voici :

```
10 REM Editeur graphique
20 BORDER 26 30 INK 0,26
40 INK 1,0
50 INK 2,6
60 INK 3,11
70 ORIGIN 0,0
80 f=1:g=2
90 CLS
100 FOR i=2 TO 5
110 |SCREENCOPY,i,1
120 NEXT i
130 MODE 1
140 PRINT TAB(33)"Entree":PRINT:PRINT
150 PRINT"Creer ou montrer image";TAB(36)"1":PRINT
160 PRINT"Charger image";TAB(36)"2":PRINT
170 PRINT"Stocker image";TAB(36)"3":PRINT
180 PRINT"Sortie de l'affectation clavier";TAB(36)"4":PRINT
190 PRINT"Contenu disque";TAB(36)"5":PRINT
200 PRINT"Fin du programme";TAB(36)"6"
210 PRINT:PRINT:INPUT"Votre choix ";a
220 IF a<1 OR a>6 THEN 130
230 IF a=6 THEN END
240 IF a=5 THEN MODE 2:CAT:GOSUB 1150:GOTO 130
```

```
250 IF a=4 THEN GOSUB 970:GOTO 130
260 PRINT:PRINT:PRINT
270 IF a=1 THEN INPUT"No de depart (image 2, 3, 4 ou 5) ";n:GOTO
290
280 INPUT"No image (Nombre entre 2 et 5) ";n
290 IF n<2 OR n>5 THEN 130
300 IF a=1 THEN 430
310 PRINT:PRINT:PRINT"Entrez un nom d'image"
320 INPUT"de 8 caracteres maximum ";bn$
330 IF LEN(bn$)<1 OR LEN (bn$)>8 THEN 130
340 IF a=2 THEN 390
350 REM stocker
360 |SCREENCOPY,1,n
370 SAVE bn$,b,49152,16384
380 GOTO 130
390 REM charger
400 LOAD bn$
410 |SCREENCOPY,n,1
420 GOTO 130
430 REM creer dessins
440 |SCREENCOPY,1,n
450 IF INKEY(0)=0 THEN y=y+g
460 IF INKEY(1)=0 THEN x=x+g
470 IF INKEY(2)=0 THEN y=y-g
480 IF INKEY(8)=0 THEN x=x-g
490 IF INKEY(28)=32 THEN g=g+1
500 IF INKEY(25)=0 THEN g=g-1:IF g<1 THEN g=1
510 IF INKEY(68)=0 THEN PLOT x,y,f
520 IF INKEY(9)=0 THEN DRAW x,y,f
530 IF INKEY(37)=0 THEN ga$="p":GOSUB 780
540 IF INKEY(52)=0 THEN ga$="d":GOSUB 780
550 IF INKEY(59)=0 THEN f=0
560 IF INKEY(60)=0 THEN f=1
570 IF INKEY(50)=0 THEN f=2
580 IF INKEY(54)=0 THEN f=3
590 IF INKEY(47)=0 THEN GOSUB 940
600 IF INKEY(65)=0 AND n<>2 THEN i=2:GOSUB 730
610 IF INKEY(57)=0 AND n<>3 THEN i=3:GOSUB 730
620 IF INKEY(56)=0 AND n<>4 THEN i=4:GOSUB 730
630 IF INKEY(49)=0 AND n<>5 THEN i=5:GOSUB 730
640 IF INKEY(6)=0 THEN |SCREENCOPY,n,1:CLEAR INPUT:GOTO 130
```

```
650 IF x<0 THEN x=0
660 IF x>639 THEN x=639
670 IF y<0 THEN y=0
680 IF y>399 THEN y=399
690 LOCATE 1,1:PRINT n
700 LOCATE 35,25:PRINT x
710 LOCATE 35,1:PRINT y
720 GOTO 450
730 REM sous-programme 'echange d'ecrans'
740 CLEAR INPUT
750 |SCREENCOPY,n,1
760 |SCREENCOPY,1,i
770 n=i:RETURN
780 REM sous-programme 'cercle'
790 |SCREENCOPY,n,1
800 CLEAR INPUT
810 CLS
820 PRINT"Entrees pour le cercle":PRINT:PRINT:PRINT
830 INPUT"Rayon (en pixels) ";r:PRINT
840 INPUT"Angle (en degres)  ";w
850 IF w>360 THEN w=360
860 |SCREENCOPY,1,n
870 DEG
880 FOR i=1 TO w
890 IF ga$="p" THEN PLOT x+r*COS(i),y+r*SIN(i),f:GOTO 920
900 PLOT x,y,f
910 DRAW x+r*COS(i),y+r*SIN(i),f
920 NEXT i
930 RETURN
940 REM sous-programme 'affectation touches'
950 |SCREENCOPY,n,1
960 CLEAR INPUT
970 CLS
980 PRINT"Touche";TAB(20)"Signification":PRINT:PRINT
990 PRINT"Touches curseur";TAB(20)"Modifier coordonnees":PRINT
1000 PRINT"+,-";TAB(20)"Modifier":PRINT TAB(20)"accelerer": PRINT
1010 PRINT"TAB";TAB(20)"Deplacer curseur":PRINT
1020 PRINT"COPY";TAB(20)"Tracer ligne":PRINT
1030 PRINT"k";TAB(20)"cercle":PRINT
1040 PRINT"g";TAB(20)"cercle plein":PRINT
1050 PRINT"w,s,r,b";TAB(20)"blanc, noir":PRINT TAB(20)"rouge,
```

```
bleu":PRINT
1060 PRINT"Espace";TAB(20)"Affectation touches":PRINT TAB(20)"et
retour a l'image":PRINT
1070 PRINT"2,3,4,5 (haut)";TAB(20)"Echange d'images":PRINT
1080 PRINT"ENTER";TAB(20)"Menu"
1090 CLEAR INPUT
1100 GOSUB 1150
1110 IF a=4 THEN RETURN
1120 |SCREENCOPY,1,n
1130 CLEAR INPUT
1140 RETURN
1150 REM sous-programme 'attendre'
1160 x$=INKEY$:IF x$="" THEN 1160
1170 RETURN
```

❐ Liste des variables :

a	Réponse 'choix du menu'
bn$	Nom d'une image
f	`crayon' des instructions graphiques
g	Vitesse de la modification de coordonnées
ga$	Variable auxiliaire indiquant si un cercle doit être rempli
i	Index de comptage et variable auxiliaire dans le sous-programme 'échange d'écrans'
n	Numéro de départ ou d'image
r	Rayon en pixels
w	Angle en degrés
x	Coordonnée x
x$	Valeur de la touche enfoncée
y	Coordonnée y

Description du programme :

10	Commentaire
20	Représentation du bord de l'écran en blanc brillant.
30-60	On affecte au numéros PEN ou PAPER 0 à 3 les couleurs blanc brillant (ligne 30), noir (ligne 40), rouge clair (ligne 50) et bleu ciel (ligne 60).
70	On détermine l'origine du curseur graphique.
80	On affecte une valeur initiale à la couleur du 'crayon' et à la vitesse.
90	Vidage de l'écran.
100-120	L'écran effacé est copié dans les quatre pages de la RAM supplémentaire de façon à effacer les signes qui pourraient s'y trouver par hasard après le premier lancement du programme.
130-200	Vidage de l'écran par le passage en mode 40 colonnes puis sortie du menu.
210-220	Entrée d'une valeur pour choisir dans le menu. Les lignes suivantes seront traitées en fonction de cette entrée.
230	Fin du programme.
240	Sortie du contenu de la disquette et retour au menu. Comme il faut absolument empêcher ici tout glissement de l'écran vers le haut, la sortie est faite en mode 80 colonnes. Il faut appuyer sur une touche avant le retour au menu (voyez le sous-programme 'attendre').
250	Sortie de l'affectation des touches par saut au sous-programme correspondant et retour au menu.

260-290	Après la sortie de trois lignes vides, une valeur (entre 2 et 5, comme les pages de la RAM supplémentaire) est demandée pour le numéro de départ ou d'image.
300-330	Si aucun dessin ne doit être créé (voir la ligne 300), on entre un nom d'image. Si ce nom est trop long (ou trop court) au retourne au menu.
340-380	Si aucune image ne doit être chargée (voyez la ligne 340), l'image de la page n de la RAM supplémentaire est copiée dans l'écran (ligne 360). L'image telle que vous la voyez est stockée de façon binaire. La mémoire écran commence à partir de l'adresse 49152 et elle occupe 16 K. On retourne ensuite au menu.
390-420	Cette section du programme n'est atteinte que si un '2' a été entré lors du choix du menu. Conformément à ce choix, une image portant le nom indiqué est maintenant chargée (ligne 400) puis copiée dans la page n de la RAM supplémentaire (ligne 410).
430-720	Section 'créer des dessins' du programme. Cette partie du programme est parcourue (voyez le retour en ligne 720) jusqu'à ce que la touche ENTER soit appuyée (voyez la ligne 640). Avant toutefois que ne commence la réalisation ou la continuation du dessin, l'image de la page n doit être copiée sur l'écran (ligne 440). Le clavier est ensuite interrogé avec INKEY. Les lignes 450-640 ont à cet égard la signification suivante :
450-480	Modification des coordonnées.
490-500	La vitesse de la modification des coordonnées peut être accélérée ou ralentie. Elle ne peut pas descendre en dessous de 1.
510	Déplacement du curseur graphique.
520	Dessin d'une ligne.

530-540	Saut au sous-programme 'cercle' et définition de la variable auxiliaire ga$.
550-580	La couleur du 'crayon' est modifiée le cas échéant.
590	Saut au sous-programme 'affectation des touches'.
600-630	La variable auxiliaire reçoit une valeur en fonction de la touche enfoncée puis on saute au sous-programme 'échange d'écrans'.
640	Après que la touche ENTER ait été actionnée, l'image est copiée dans la page n, le buffer clavier est vidé et on retourne au menu. Les lignes 650-710 vérifient alors les coordonnées x et y et les corrigent le cas échéant en fonction des pixels disponibles. Les valeurs des variables n, x et y sont sorties dans les coins de l'écran. Si vous le souhaitez, vous pouvez naturellement ajouter encore d'autres instructions INKEY. Il serait par exemple intéressant d'employer les instructions FILL ou MASK de manière judicieuse.
730-770	Sous-programme 'échange d'écrans'. Après que le buffer clavier ait été vidé (ligne 740), la page actuelle (page 1) est copiée dans la page n (ligne 750) et la page i apparaît sur l'écran (ligne 760). La variable i a été définie avant le saut à ce sous-programme. Elle est identique au numéro de l'image voulue. Comme l'image i est maintenant l'image actuelle, n est fixé égal à i avant le retour.
780-930	Sous-programme 'cercle'. Avant les entrées nécessaires pour définir le cercle, l'image actuelle est à nouveau copiée dans la RAM supplémentaire (ligne 790). L'image actuelle apparaît ensuite à nouveau sur l'écran (ligne 860) et le cercle est dessiné en fonction des valeurs entrées (lignes 870-920). La ligne 890 décide d'après la variable ga$ s'il faut ou non dessiner un cercle plein. Pour un cercle plein, on trace des lignes du centre de l'écran aux différents points du cercle (voyez les lignes 900-910) alors qu'on se contente sinon de plotter les points du cercle (ligne 890).

940-1140	Sous-programme 'affectation de touches'. Si ce sous-programme est appelé à partir du menu, il commence à la ligne 970 et il est abandonné en ligne 1110. Les instructions SCREEN, CLEAR et INPUT ne sont donc exécutées que si le programme est appelé pendant la réalisation d'une image.
1150-1170	Sous-programme 'attendre'.

7. La musique

7.1. Introduction

Sur le plan des particularités musicales, votre CPC 6128 appartient une fois encore à la crème des ordinateurs familiaux. En tant que possesseur d'un CPC, vous n'êtes pas cantonné à la production de bruits comme c'est le cas avec quelques ordinateurs de même catégorie. Vous pouvez même, au contraire, avec un peu d'exercice, imiter différents instruments et même programmer et faire exécuter une mélodie complète avec son harmonie. D'ailleurs, si le haut-parleur intégré suffit à l'écoute de vos programmes musicaux, la connexion de votre ordinateur à une chaîne stéréo est recommandée si vous voulez tirer pleinement parti des possibilités musicales du CPC.

Naturellement, la programmation musicale n'est pas l'objet principal de cet ouvrage mais nous allons néammoins vous expliquer dans le chapitre suivant les principales notions ainsi que les différentes instructions permettant de produire des bruits et des sons. Même si vous ne vous intéressez pas le moins du monde à la musique, ce chapitre peut malgré tout retenir votre attention. La musique informatique relève en effet essentiellement du domaine des mathématiques et de la physique. Si dans ces conditions vous pensez que ce sujet n'est pas pour vous, sachez toutefois que tel ou tel programme n'ayant rien à voir avec la musique pourra être parfois nettement embelli par une musique d'accompagnement.

7.2. Editeur musical de base

Si vous examinez les différents paramètres des instructions musicales SOUND, ENV et ENT, alors vous constaterez que ces instructions se distinguent assez nettement de celles que nous avons rencontrées jusqu'ici. Alors que beaucoup des instructions que nous avons vues jusqu'ici peuvent être employées avec des paramètres répétitifs, la programmation

musicale nécessite un certain effort d'adaptation. En effet les noms des paramètres sont déjà quelque chose de déroutant et d'inhabituel pour les lecteurs qui n'ont pas d'expérience en la matière.

C'est pourquoi, pour ne pas vous décourager d'emblée avec l'explication des différentes instructions musicales, nous allons vous présenter maintenant un petit programme d'éditeur musical qui contribuera, nous l'espérons, à éveiller votre intérêt pour la programmation musicale.

La structure du programme est aussi simple que possible car il travaille uniquement avec les deux premiers paramètres de l'instruction SOUND. Le premier paramètre 'état canal' n'est, qui plus est, pas modifié et il conserve toujours la valeur '1', c'est-à-dire que nous utilisons pour tous les sons produits le premier des trois canaux sonores disponibles. Seul le second paramètre 'période de note' varie. On peut choisir pour ce paramètre n'importe quelles valeurs entre 0 et 4095 mais seules quelques-unes de ces valeurs fournissent des notes 'musicales'. Ces valeurs sont toutes énumérées dans votre manuel d'utilisation. Le programme suivant utilise les valeurs de l'octave 0. Si vous voulez utiliser une autre octave, vous n'avez pas besoin d'entrer à nouveau les valeurs correspondantes car vous pouvez calculer ces valeurs de note à l'aide de la formule suivante :

Valeur de note = valeur de note de la note de l'octave 0 / 2^octave

On obtiendra par exemple pour la note do de la seconde octave : 478/2^2

La ligne de programme supplémentaire à rajouter pourrait se présenter ainsi :

```
165 t=t/2^<numéro d'octave>
```

Comme cette équation peut aisément engendrer des erreurs d'arrondissement, il est conseillé de travailler avec l'équation présentée au chapitre 7 de votre manuel d'utilisation. Quelle que soit la formule que vous utilisez, vous pouvez bien sûr

également entrer au début du programme l'octave voulue, avec une instruction INPUT.

Le point fort du programme réside dans le fait que les notes sont produites en frappant sur des touches. Comparé à l'entrée directe d'instructions SOUND, cela représente un allègement considérable du travail. L'affectation du clavier correspond aux noms de notes allemands qui présentent en effet le grand avantage de n'être composés que d'une lettre, chaque lettre ne correspondant bien sûr qu'à une seule note alors que, dans le système français, le sol et le si commencent tous deux par 's'. Notez par ailleurs que la note dièse correspondant à chaque note naturelle est obtenue avec SHIFT. La table suivante vous indique la correspondance des touches utilisées par le programme avec les noms de notes français :

Touche	Signification
c	Do
C	Do dièse
d	Ré
D	Ré dièse
e	Mi
f	Fa
F	Fa dièse
g	Sol
G	Sol dièse
a	La
A	La dièse
b	Si

Outre la production des sons, le programme se charge également d'indiquer sur l'écran quelle touche a été appuyée. Cela facilitera aux 'compositeurs' le contrôle de la mélodie jouée. Le fait d'actionner une touche non énumérée dans la table ci-dessus n'a aucun effet sur le déroulement du programme. Seule la touche ESC a pour fonction d'interrompre le programme.

```
10 REM Editeur musical de base
20 t$=INKEY$
30 IF t$="c" THEN t$="c Do":t=239:GOTO 160
40 IF t$="C" THEN t$="C Do #":t=225:GOTO 160
50 IF t$="d" THEN t$="d Re":t=213:GOTO 160
60 IF t$="D" THEN t$="D Re #":t=201:GOTO 160
70 IF t$="e" THEN t$="e Mi":t=190:GOTO 160
80 IF t$="f" THEN t$="f Fa":t=179:GOTO 160
90 IF t$="F" THEN t$="F Fa #":t=169:GOTO 160
100 IF t$="g" THEN t$="g Sol":t=159:GOTO 160
110 IF t$="G" THEN t$="G Sol #":t=150:GOTO 160
120 IF t$="a" THEN t$="a La":t=142:GOTO 160
130 IF t$="A" THEN t$="A La #":t=134:GOTO 160
140 IF t$="b" THEN t$="b Si":t=127:GOTO 160
150 GOTO 20
160 PRINT t$
170 SOUND 1,t
180 GOTO 20
```

❐ Description du programme :

10	Commentaire.
20	Interrogation du clavier.
30-140	Evaluation de la touche enfoncée. Le paramètre 'période de note' de l'instruction SOUND est défini en fonction de la touche appuyée. On saute ensuite à la sortie sur écran (ligne 160) et à la production du son (ligne 170).
150	Retour à l'interrogation du clavier si aucune touche de la table d'affectation n'a été enfoncée.
160	Sortie de la note sur l'écran.
170	Production du son au moyen de l'instruction SOUND.
180	Retour à l'interrogation du clavier.

Il y a diverses possibilités d'améliorer et compléter le programme. Voici deux propositions :

❶ Sauvegarde et chargement de la mélodie programmée

❷ Utilisation des autres canaux musicaux (voyez le chapitre suivant)

7.3. Sound et Rendez-vous

SOUND est la principale instruction de production de sons. Rien n'est possible sans elle, c'est-à-dire qu'il n'est pas possible de produire de sons. L'instruction SOUND a au total 7 paramètres. Alors que les cinquième et sixième paramètres concernent les courbes d'enveloppe, qui seront expliquées plus en détail au chapitre suivant, les paramètres 2, 3, 4 et 7 sont très faciles à comprendre. Leur signification a été évoquée en partie ou peut du moins parfaitement être comprise en s'aidant du manuel d'utilisation.

Dans ce chapitre, notre intérêt portera surtout sur le premier paramètre, l'état de canal. Cette valeur indique à votre CPC quel canal de son il doit utiliser et s'il doit synchroniser ce son avec la séquence de sons des autres canaux ou s'il doit éventuellement tenir ce son. Le fait de synchroniser les séquences de notes sur les trois canaux est appelé 'rendez-vous'. Un rendez-vous permet par exemple de faire jouer une mélodie avec des pauses sans que cela ne produise de 'claquements' du haut-parleur.

Comment peut-on se représenter, de façon symbolique, l'arrangement d'un rendez-vous ?

Vous savez que votre CPC dispose de 3 canaux et qu'il peut stocker pour chaque canal jusqu'à 5 notes programmées, c'est-à-dire qu'il peut placer ces notes dans une position d'attente. Un rendez-vous entre deux canaux aura pour effet que les notes prévues pour les deux canaux ne seront jouées que lorsque les deux canaux seront libres.

Que vous ayez compris d'emblée ce principe ou non, vous êtes certainement désireux d'en faire l'expérience. Il vous faut pour cela connaître toutefois les valeurs de l'état canal correspondant aux différentes activités que vous pouvez provoquer. Ces valeurs sont calculées à partir de l'octet suivant :

Bit	Valeur	Instruction
0	1	Son envoyé au canal A
1	2	Son envoyé au canal B
2	4	Son envoyé au canal C
3	8	Rendez-vous avec le canal A
4	16	Rendez-vous avec le canal B
5	32	Rendez-vous avec le Canal C
6	64	Arrêt du canal
7	128	Effacement du canal

Dans l'éditeur de son du dernier chapitre, nous avons toujours utilisé le canal A, nous avons donc toujours pris 1 comme valeur pour le canal. Si nous voulons par exemple utiliser le canal B, nous fixerons 2 comme valeur et si nous voulons envoyer une note en même temps aux canaux A et B, il nous suffit d'additionner les valeurs correspondantes et nous obtiendrons la valeur 3. Ce principe de l'addition de valeurs différentes est appliqué chaque fois qu'il s'agit de combiner les différentes valeurs indiquées ci-dessus. Le numéro de canal 10 indiquera donc qu'une note sera jouée sur le canal B et qu'on devra attendre une note prévue sur le canal A pour un rendez-vous avec elle. Vous pouvez donc effectuer de cette manière toutes les combinaisons voulues. Si vous additionnez toutes les valeurs, vous obtenez le nombre 255. Il n'y pas d'autres instructions justifiant des valeurs plus élevées (voir le chapitre 4).

Indiquons encore ce qu'on entend par 'arrêt du canal' et 'effacement du canal'. 'Arrêt' d'une note signifie que la note correspondante ne sera jouée qu'après l'instruction RELEASE alors que l'effacement entraîne un arrêt brusque de la note actuellement jouée par le canal considéré.

Mais revenons-en aux différentes valeurs pour le canal. Comme vous l'avez peut-être déjà remarqué, la valeur du canal est toujours égale à 2^bit. Si vous combinez les différents bits, vous obtenez un nombre binaire puisque la valeur d'un bit est 1 ou 0. C'est ainsi par exemple que la valeur 10 équivaut au nombre binaire 1010, c'est-à-dire à :

```
1*2^3+0*2^2+1*2^1+0*2^0.
```

Ce n'est pas le lieu ici de réexpliquer le principe du système binaire (voyez le chapitre 4.4). Mais comme vous l'aviez déjà vu avec l'instruction SYMBOL, même en tant que programmeur BASIC, il vous est difficile d'ignorer le principe des nombres binaires. L'étude des nombres binaires est donc particulièrement conseillée, surtout si vous voulez mieux comprendre le principe du fonctionnement de votre ordinateur.

Une bonne compréhension du programme suivant nécessite également des connaissances de base sur le système binaire. Ce programme vous offre en effet une aide pour le maniement de l'état canal qui n'est certainement pas très facile au début. Vous pouvez choisir dans le cours du programme entre deux variantes. La première possibilité est d'indiquer une valeur pour l'état canal. Vous obtenez alors comme résultat les instructions exprimées par cette valeur de canal. L'autre possibilité consiste à partir des instructions possibles et à demander la valeur canal correspondant à cette combinaison d'instructions. Si vous n'êtes donc pas encore très assuré dans le maniement des valeurs de canal, il vous suffit de taper le programme et de l'essayer. Vous verrez que cela en vaut la peine.

```
10 REM Aide au calcul de la valeur de canal
20 DATA Son envoye au canal A,Son envoye au canal B,Son envoye au
canal C
30 DATA Rendez-vous avec le canal A,Rendez-vous avec le canal
B,Rendez-vous avec le canal C
40 DATA Arret de la note,Effacement de la file d'attente
50 FOR i=0 TO 7
60 READ b$(i):NEXT i
70 CLS
```

```
80 PRINT"S'il faut partir d'une valeur precise"
90 PRINT"pour l'etat canal (le programme"
100 PRINT"determinera alors les fonctions"
110 PRINT"correspondantes), entrez un 1.":PRINT
120 PRINT"Si le programme doit trouver la"
130 PRINT"valeur de l'etat canal pour les"
140 PRINT"fonctions que vous entrerez, entrez 2."
150 PRINT:PRINT:PRINT:INPUT"Votre choix ";a
160 IF a<>1 AND a<>2 THEN 150
170 CLS:IF a=2 THEN 240
180 INPUT"Etat canal ";k:PRINT:PRINT
190 IF k<1 OR k>255 THEN 180
200 FOR i=7 TO 0 STEP-1
210 IF k>=2^i THEN k=k-2^i:PRINT:PRINT b$(i)
220 NEXT i
230 GOTO 320
240 PRINT"Repondez '1' pour 'oui'"
250 PRINT" et '0' pour 'non'":PRINT:PRINT
260 FOR i=0 TO 7
270 PRINT b$(i),:LOCATE 34,i+5:INPUT a
280 IF a<> 0 AND a<>1 THEN 270
290 IF a=1 THEN k=k+2^i
300 NEXT i
310 PRINT:PRINT:PRINT:PRINT"Etat canal =";k
320 PRINT:PRINT:END
```

❒ Liste de variables :

a	Variable de réponse
b$(i)	Les 8 instructions pouvant être exprimées par la valeur canal
i	Index de comptage (i s'analyse ici bit par bit)
k	Valeur de canal

Description du programme :

10	Commentaire.
20-60	Les instructions pouvant être exprimées par la valeur canal sont mises à disposition du programme et lues par celui-ci sous forme de données.
70-140	L'écran est vidé et un commentaire est sorti expliquant le mode d'emploi du programme.
150-170	Entrée d'une valeur concernant le déroulement voulu du programme. Vidage de l'écran et éventuellement saut dans le programme.
180-230	A partir de la valeur canal entrée (entre 1 et 255, voir lignes 180 -190) la boucle de programme (lignes 200-220) examine quelles puissances de 2 (2^i avec i=7 à i=0) contient la valeur canal. Si la valeur pour k contient une puissance de 2, k est diminué de cette puissance de 2 et l'instruction correspondante est sortie (ligne 210). La ligne 230 saute alors à la fin du programme.
240-310	Après une sortie de commentaire (lignes 240-250), la boucle de programme (lignes 260-300) sort les 8 instructions possibles et demande une valeur en réponse à la question de savoir si l'instruction présentée est souhaitée ou non (ligne 270). Chaque fois qu'il a été répondu 'oui', la valeur canal est augmentée de la puissance de 2 correspondante (ligne 290). Une fois la boucle terminée, le résultat est sorti (ligne 310).
320	Fin du programme.

Une fois que vous aurez fait, avec plus ou moins de réussite, vos premiers essais avec ce programme, vous voudrez certainement constater directement l'effet des résultats de vos exercices. Vous avez cependant certainement constaté qu'il est assez compliqué de fixer l'une après l'autre différentes instructions SOUND. C'est pourquoi nous vous fournissons un petit programme qui vous dégage d'une partie de ce travail.

Une fois le programme lancé, vous n'aurez qu'à entrer les paramètres des différentes instructions SOUND. Notez cependant que nous nous sommes limités aux trois premiers paramètres. Vous pouvez bien sûr déterminer vous-même le nombre d'instructions. Et pour que vous puissiez conserver une vue d'ensemble, une fenêtre a été définie pour l'entrée des paramètres ainsi que deux autres fenêtres pour la sortie des 50 dernières instructions. Voici donc un exemple supplémentaire d'emploi très pratique de la technique des fenêtres.

Après que toutes les instructions aient été entrées, elles sont jouées les unes après les autres.

```
10 REM Aide pour SOUND
20 MODE 2
30 WINDOW #0,1,19,1,25:WINDOW #1,21,50,1,25:WINDOW #2,51,80,1,25
40 PRINT"Nombre d'ordres":INPUT"SOUND ";n
50 DIM k(n),t(n),d(n)
60 PRINT:PRINT:PRINT:FOR i=1 TO n
70 PRINT "Instruction"i:PRINT
80 INPUT"Canal ";k(i)
90 INPUT"Note  ";t(i)
100 INPUT"Duree ";d(i)
110 IF j=1 THEN j=2 ELSE j=1
120 PRINT#j,"SOUND";k(i);",";t(i);",";d(i)
130 PRINT:PRINT:NEXT i
140 FOR i=1 TO n
150 SOUND k(i),t(i),d(i)
160 NEXT i
170 PRINT:PRINT:PRINT:PRINT"Jouer encore":INPUT"une fois (o/n)
";a$
180 IF a$="o" THEN 140
190 IF a$="n" THEN MODE 1 ELSE GOTO 170
```

Liste de variables :

a$	Chaîne de réponse (o/n)
d(i)	Durée
i	Index de comptage
j	Paramètre de l'instruction WINDOW (1 ou 2)
k(i)	Canal
n	Nombre d'instructions SOUND
t(i)	Période de note

Description du programme :

10	Commentaire.
20	Passage en mode 80 colonnes.
30	Définition de trois fenêtres écran. Alors que la fenêtre 0 servira aux entrées, les fenêtres 1 et 2 seront utilisées pour la sortie des instructions SOUND définies.
40	Entrée du nombre d'instructions SOUND.
50	Réservation de place mémoire pour les trois premiers paramètres.
60-130	Pour chacune des n instructions SOUND, les trois premiers paramètres sont définis par des valeurs entrées par l'utilisateur. Les entrées interdites ne sont pas interceptées. En lignes 110-120, les instructions SOUND définies sont sorties à tour de rôle (ligne 110) dans la première ou la seconde fenêtre. La partie droite de l'écran doit donc être lue ligne par ligne de gauche à droite. Un maximum de 50 instructions SOUND peuvent donc être ainsi présentées en même temps.
140-160	Exécution des instructions SOUND.

170-190	On demande si les instructions doivent être rejouées. Un saut se produit dans le programme si cela est souhaité. Sinon le programme est terminé.

A la fin de ce chapitre, voici encore deux bonnes idées, l'une concernant le rendez-vous et l'autre la connexion de votre ordinateur sur une chaîne stéréo.

Si vous avez arrangé un rendez-vous en suivant la méthode indiquée plus haut, aucune note ne peut plus entrer dans le canal avant exécution du rendez-vous. Si vous voulez éviter cela, vous avez la possibilité de fixer une instruction d'arrêt pour retarder l'exécution d'un canal plein. L'instruction RELEASE vous permet ensuite d'appeler à nouveau ce canal. Vous disposez ainsi d'une autre possibilité intéressante pour la synchronisation des canaux. Notez que le paramètre de l'instruction RELEASE est aussi un bit dont la signification s'analyse bit par bit.

Pour relier des canaux différents, vous pouvez reprendre les trois premières instructions de la table que nous vous avons donnée plus haut pour l'instruction SOUND. Evoquons encore rapidement à cet égard la fonction SQ dont nous n'avons pas parlé jusqu'à présent. Cette fonction renvoie le nombre de places libres dans la file d'attente du canal indiqué. La valeur sortie par cette fonction s'analyse également bit par bit (voyez le manuel d'utilisation). Si vous voulez notamment éviter que le déroulement du programme ne soit arrêté en raison d'une file d'attente pleine, l'emploi de la fonction SQ ou ON SQ GOSUB est particulièrement recommandé.

Si vous connectez votre ordinateur à une chaîne stéréo, vous pourrez vraiment exploiter pleinement toutes les possibilités techniques offertes par la programmation musicale. Voici dans ce cas la répartition des trois canaux de son sur les deux canaux stéréo: alors que les sons du canal A et du canal C sont envoyés sur un canal stéréo différent, les sons du canal B apparaissent sur les deux canaux stéréo. Avec un travail de programmation approprié, il n'est donc pas difficile d'obtenir des effets stéréo.

7.4. Enveloppe de volume et de hauteur de note

Les instructions de courbes d'enveloppe ENV et ENT sont conçues pour modifier un son produit par l'instruction SOUND, par exemple pour imiter un instrument de musique. Alors que l'instruction ENV permet de modifier le volume d'une note (en l'élevant ou en l'abaissant), l'instruction ENT sert à réaliser une modification minime de la hauteur de note.

Comme ces deux instructions sont expliquées de manière très complète dans votre manuel d'utilisation, nous n'allons pas les réexpliquer ici. Nous allons plutôt nous limiter, dans les développements suivants, à quelques notions de base sur ces deux instructions pour vous présenter ensuite deux programmes que l'on peut vraiment qualifier de programmes utilitaires ou d'enseignement en ce qui concerne les courbes d'enveloppe.

Chaque instruction de courbe d'enveloppe a pour paramètres le 'numéro de courbe d'enveloppe', le 'nombre de pas', 'l'amplitude du pas' et la 'durée du pas'. Le numéro de courbe d'enveloppe est un nombre qui sert à identifier la courbe d'enveloppe définie pour l'instruction SOUND. Le nombre de pas détermine combien de modifications de volume ou de hauteur de note un son devra parcourir, l'amplitude du pas indique la différence entre les différents pas et la durée de pas indique la durée d'un son pour chaque pas. Votre manuel d'utilisation indique quelles valeurs sont autorisées pour chacun de ces paramètres.

Cependant, vous pouvez également combiner pour une note la montée et la chute du volume ou de la hauteur de note. Vous pouvez en effet diviser une courbe d'enveloppe en jusqu'à 5 sections différentes. Pour chacune de ces sections, vous pouvez définir indépendamment les trois paramètres 'nombre de pas', 'amplitude du pas' et 'durée du pas'. Suivant le nombre de sections qui doivent être définies, les instructions de courbe d'enveloppe ont donc au moins trois paramètres et au plus 16 paramètres. Pour l'instruction ENV, il suffira cependant le plus souvent de définir trois sections, à savoir l'attaque (montée du volume), la tenue (maintien du volume) et le relâchement

(baisse du volume). Si vous voulez par exemple imiter un instrument à cordes, le relâchement dans l'instruction ENV devra être plus long que l'attaque car les instruments à corde font en général entendre un petit écho dû à la vibration des cordes.

Pour adapter les instructions de courbe d'enveloppe aux instructions SOUND correspondantes, il est important de connaître la longueur totale d'une courbe d'enveloppe (en centièmes de seconde). Celle-ci est donnée par le produit du nombre de pas (nombre d'étapes) et de la durée (durée par étape). Si vous avez défini plusieurs sections, vous devez additionner les résultats obtenus pour chaque section. Notez que pour l'instruction ENV il est possible de ne pas indiquer le paramètre 'durée' dans l'instruction SOUND correspondante, étant donné que la courbe d'enveloppe de volume détermine également la longueur d'une note. Cela ne vaut pas toutefois pour la courbe d'enveloppe de hauteur de note.

Vous trouverez dans votre manuel d'utilisation une feuille vous permettant de dessiner les courbes (vibrations carrées serait un terme plus précis) pour vos courbes d'enveloppe. Pour bien comprendre les courbes d'enveloppe, il est cependant préférable de faire tout simplement différents essais avec les paramètres. Pour vous aider en cela et pour vous donner goût à ce type de travail, voici deux programmes d'aide et d'apprentissage. Le second programme ne constitue qu'une version complétée du programme d'aide pour l'instruction SOUND que nous vous donnions au dernier chapitre. C'est pourquoi nous n'en dirons pas plus ici (voyez la description du programme). Le programme que voici maintenant concerne par contre exclusivement l'instruction ENV. En y apportant les petites modifications nécessaires, il peut aussi être utilisé pour l'instruction ENT. Nous avons choisi ici l'instruction ENV car c'est elle qui permet d'obtenir le plus grand éventail de modifications du son. Ce programme présente en outre une combinaison intéressante des instructions INK, KEY DEF, LOCATE et INKEY$, de sorte qu'il a véritablement un caractère exemplaire y compris pour d'autres applications.

Comme le programme est commandé par une interrogation du clavier (INKEY$) et qu'il ne sort pas en principe d'explications sur le déroulement du programme pour ne pas modifier l'image (ou plus précisément le tableau) que vous voyez sur l'écran après le lancement du programme, il est nécessaire que nous vous donnions ici une brève énumération des entrées possibles. Vous devez pour cela vous représenter la commande du programme à deux niveaux différents. Au premier niveau, vous pouvez déterminer pour quelle section de courbe d'enveloppe les paramètres doivent être fixés ou modifiés. Vous disposez à cet effet des touches "1", "2", "3", "4" et "5". Dès que vous avez appuyé sur une de ces touches, elle clignote pour vous indiquer la section de courbe d'enveloppe actuellement traitée.

Vous disposez en outre à ce niveau de la possibilité d'appeler les paramètres 'nombre de pas' (entrée "n"), 'amplitude du pas' (entrée "a") ou 'durée du pas' (entrée "d") pour les modifier. Le paramètre actuel clignote également. L'entrée d'une de ces trois lettres entraîne un saut au second niveau.

Vous avez alors la possibilité d'élever ou d'abaisser la valeur du paramètre considéré pour la section de courbe d'enveloppe actuelle en actionnant la touche 'curseur haut' ou 'curseur bas'. La technique de programmation employée fait que seules des valeurs autorisées peuvent apparaître. Vous pouvez quitter le second niveau en appuyant simplement sur la touche RETURN ou ENTER. Vous vous retrouvez alors à nouveau dans le premier niveau où vous avez maintenant la possibilité de faire retentir la courbe d'enveloppe (en liaison avec une instruction SOUND que vous pouvez modifier, voir le programme) en actionnant la touche espace. Vous pouvez ensuite mettre fin au programme ou bien entreprendre d'autres modifications. Un message approprié vous l'indique à cet endroit.

Peut-être cette explication du mode d'emploi du programme vous a-t-elle semblée tellement compliquée que vous n'avez plus aucune envie de taper le listing. Mais ne vous laissez pas égarer. Vous parviendrez vite à maîtriser l'emploi des touches correspondantes pour pouvoir vous concentrer sur le 'travail' véritable. Amusez-vous bien.

```
10 REM Aide et editeur pour ENV
20 CLS
30 LOCATE 16,1:PRINT"Sections de courbe"
40 LOCATE 16,3:PRINT"1    2    3    4    5"
50 PRINT:PRINT:PRINT"Nombre de pas":PRINT:PRINT "Amplitude":
PRINT:PRINT"Duree de pas"
60 KEY DEF 0,1,104
70 KEY DEF 2,1,98
80 INK 2,24,1
90 ha=1
100 a$=INKEY$:IF a$="" THEN 100
110 IF a$="1" OR a$="2" OR a$="3" OR a$="4" OR a$="5" THEN LOCATE
10+5*ha,3:PRINT ha:ha=VAL(a$):PEN 2:LOCATE 10+5*ha,3:PRINT ha:PEN 1
120 IF a$="n" THEN GOSUB 280
130 IF a$="a" THEN GOSUB 350
140 IF a$="d" THEN GOSUB 420
150 IF a$=" " THEN 170
160 GOTO 100
170 td=0
180 FOR i=1 TO 5
190 td=td+sa(i)*sz(i)
200 NEXT i
210 ENV 1,sa(1),sw(1),sz(1),sa(2),sw(2),sz(2),sa(3),sw(3),sz(3),
sa(4),sw(4),sz(4),sa(5),sw(5),sz(5)
220 SOUND 1,478,0,0,1
230 WINDOW #1,1,40,17,25
240 PRINT#1,"Duree de note (en 100emes de sec):";td
250 PRINT#1,:PRINT#1,"Appuyez sur espace pour continuer. Toute
autre touche entraine l'arret du programme."
260 a$=INKEY$:IF a$="" THEN 260
270 IF a$=" " THEN CLS#1:GOTO 100 ELSE LOCATE 1,24:END
280 REM SP Modifier nombre de pas
290 PEN 2:LOCATE 1,6:PRINT"Nombre de pas":PEN 1
300 a$=INKEY$:IF a$="" THEN 300
310 IF a$="h" AND sa(ha)<127 THEN sa(ha)=sa(ha)+1:LOCATE
10+5*ha,6:PRINT sa(ha)
320 IF a$="b" AND sa(ha)>0 THEN sa(ha)=sa(ha)-1:LOCATE
10+5*ha,6:PRINT sa(ha)
330 IF a$=CHR$(13) THEN LOCATE 1,6:PRINT"Nombre de pas":RETURN
340 GOTO 300
350 REM SP Modifier amplitude du pas
```

```
360 PEN 2:LOCATE 1,8:PRINT"Amplitude":PEN 1
370 a$=INKEY$:IF a$="" THEN 370
380 IF a$="h" AND sw(ha)<127 THEN sw(ha)=sw(ha)+1:LOCATE
10+5*ha,8:PRINT sw(ha)
390 IF a$="b" AND sw(ha)>-128 THEN sw(ha)=sw(ha)-1:LOCATE
10+5*ha,8:PRINT sw(ha)
400 IF a$=CHR$(13) THEN LOCATE 1,8:PRINT"Amplitude":RETURN
410 GOTO 370
420 REM SP Modifier duree du pas
430 PEN 2:LOCATE 1,10:PRINT"Duree":PEN 1
440 a$=INKEY$:IF a$="" THEN 440
450 IF a$="h" AND sz(ha)<255 THEN sz(ha)=sz(ha)+1:LOCATE
10+5*ha,10:PRINT sz(ha)
460 IF a$="b" AND sz(ha)>0 THEN sz(ha)=sz(ha)-1:LOCATE
10+5*ha,10:PRINT sz(ha)
470 IF a$=CHR$(13) THEN LOCATE 1,10:PRINT"Duree":RETURN
480 GOTO 440
```

❒ Liste de variables :

a$	Chaîne renvoyée par l'ordinateur
ha	Section de courbe d'enveloppe
i	Index de comptage
sa(i)	Nombre de pas
sw(i)	Amplitude de pas
sz(i)	Durée de pas
td	Durée de note en centièmes de seconde

❒ Description du programme :

10	Commentaire.
20	Vidage de l'écran.
30-50	Sortie des titres de tableau.

60-70	Affectation de la lettre "h" à la touche curseur haut et de la lettre "b" à la touche curseur bas.
80	Affectation au second crayon de couleur des couleurs jaune clair et bleu (clignotant).
90	La variable 'section de courbe d'enveloppe' est définie avec 1 comme valeur initiale.
100-160	Interrogation du clavier. Lorsqu'est entré un nombre entre 1 et 5 (ligne 110), la valeur actuelle de la section de courbe d'enveloppe est modifiée en conséquence (ha=VAL(a$)) et la valeur actuelle est sortie en clignotement (PEN 2) dans le titre du tableau (LOCATE 10+5*ha) après que la valeur actuelle précédente pour 'ha' ait été remplacée par le numéro de crayon de couleur 1 (les deux premières instructions après THEN). Dans les lignes 120-140 est effectué un saut aux sous-programmes correspondants suivant qu'il s'agit de modifier le nombre de pas (ligne 120), l'amplitude du pas (ligne 130) ou la durée du pas (ligne 140) pour la section de courbe d'enveloppe actuelle. Si on appuie sur la touche espace, on saute à la sortie de la musique (ligne 150). La ligne 160 sert à l'interrogation 'permanente' du clavier (si les touches des lignes 110-150 n'ont pas encore été enfoncées).
170-200	Calcul de la durée de la note fixée par l'instruction ENV.
210-220	Exécution de l'instruction ENV en liaison avec une instruction SOUND que l'utilisateur du programme peut naturellement également modifier.
230	Définition d'une fenêtre écran.
240-250	La durée de note (voir les lignes 170-200) et un message pour la poursuite du programme sont sortis dans la fenêtre définie.

260-270	Suivant la touche enfoncée, la fenêtre définie est vidée et le programme continue ou bien le programme se termine.
280-340	Sous-programme de modification du nombre de pas pour la section de courbe d'enveloppe actuelle. La ligne 290 sort le titre de tableau 'nombre de pas' en clignotement. Cela est à nouveau annulé avant le retour du sous-programme (voir la ligne 330). Les lignes 300-340 interrogent à nouveau le clavier. Si la touche 'curseur haut' est enfoncée (ligne 310), le paramètre 'nombre de pas' est chaque fois augmenté de 1 et la nouvelle valeur pour le nombre de pas est inscrite dans la table. La valeur de ce paramètre ne peut dépasser 127. On procède de même lorsqu'on appuie sur la touche 'curseur bas' (ligne 320, diminution de la valeur du paramètre). Le fait d'actionner la touche RETURN ou ENTER entraîne en ligne 330 le retour du sous-programme. La ligne 340 sert à l'interrogation 'permanente' du clavier.
350-480	Sous-programmes de modification de l'amplitude de pas et de la durée de pas (chaque fois pour la section de courbe d'enveloppe actuelle). On procède ici suivant le même principe que dans le programme précédent.

Comme nous vous l'avons déjà annoncé plus haut, voici maintenant directement la version complétée du programme d'aide à SOUND du dernier chapitre :

```
10 REM Aide pour les courbes d'enveloppe et pour SOUND
20 MODE 2
30 WINDOW #0,1,25,1,25:WINDOW #1,26,80,1,25
40 PRINT"Nombre d'ordres":INPUT"END ";n1:PRINT
50 FOR i=1 TO n1
60 GOSUB 430
70 ENV hn,sa(1),sw(1),sz(1),sa(2),sw(2),sz(2),sa(3),sw(3),
sz(3), sa(4),sw(4),sz(4),sa(5),sw(5),sz(5)
80 PRINT#1,"ENV,";hn;
90 FOR i1=1 TO aa
```

```
100 PRINT#1,",";sa(i1);",";sw(i1);",";sz(i1);
110 NEXT i1:PRINT#1
120 NEXT i:PRINT:PRINT
130 PRINT"Nombre d'ordres":INPUT"ENT ";n1:PRINT
140 FOR i=1 TO n1
150 GOSUB 430
160 ENT hn,sa(1),sw(1),sz(1),sa(2),sw(2),sz(2),sa(3),sw(3)
,sz(3), sa(4),sw(4),sz(4),sa(5),sw(5),sz(5)
170 PRINT#1,"ENT,";hn;
180 FOR i1=1 TO aa
190 PRINT#1,",";sa(i1);",";sw(i1);",";sz(i1);
200 NEXT i1:PRINT#1
210 NEXT i:PRINT:PRINT
220 PRINT"Nombre d'ordres":INPUT"SOUND ";n
230 DIM k(n),t(n),d(n),l(n),lh(n),th(n),gp(n)
240 PRINT:PRINT:FOR i=1 TO n
250 PRINT "Instruction"i:PRINT
260 INPUT"Canal           ";k(i)
270 INPUT"Note            ";t(i)
280 INPUT"Duree           ";d(i)
290 INPUT"Volume          ";l(i)
300 PRINT"Courbe d'env."
310 INPUT"de volume       ";lh(i)
320 INPUT"Courbe de note";th(i)
330 PRINT"Periode de"
340 INPUT"bruit           ";gp(i)
350 PRINT#1,"SOUND";k(i);",";t(i);",";d(i);",";
l(i);",";lh(i);","; th(i);",";gp(i)
360 PRINT:PRINT:NEXT i
370 FOR i=1 TO n
380 SOUND k(i),t(i),d(i),l(i),lh(i),th(i),gp(i)
390 NEXT i
400 PRINT:PRINT:PRINT:PRINT"Jouer encore":INPUT"une fois (o/n)
";a$
410 IF a$="o" THEN 370
420 IF a$="n" THEN MODE 1:END ELSE GOTO 400
430 REM SP Entrees de courbes d'enveloppe
440 PRINT"Numero de":INPUT"courbe ";hn
450 PRINT"Combien de":INPUT"sections ";aa
460 FOR i1=1 TO aa
470 PRINT "Section";i1;":"
```

```
480 PRINT"Pas:"
490 INPUT"-nombre ";sa(i1)
500 INPUT"-amplit.";sw(i1)
510 INPUT"-durée  ";sz(i1)
520 NEXT i1
530 RETURN
```

❐ Liste de variables :

aa	Nombre des sections de courbe d'enveloppe
a$	Chaîne de réponse (o,n)
d(i)	Durée
i	Index de comptage
i1	Index de comptage
gp(i)	Période de bruit
hn	Numéro de courbe d'enveloppe
k(i)	Canal
l(i)	Volume
lh(i)	Courbe d'enveloppe de volume
n	Nombre d'instructions SOUND
n1	Nombre d'instructions ENV ou ENT
t(i)	Période de note
th(i)	Courbe d'enveloppe de note
sa(i)	Nombre de pas
sw(i)	Amplitude de pas
sz(i)	Durée de pas

❐ Description du programme :

10	Commentaire.
20	Passage en mode 80 colonnes.
30	Définition de deux fenêtres écran. La fenêtre 0 sert à l'entrée des données et la fenêtre 1 à la sortie des instructions de courbe d'enveloppe et des instructions SOUND définies.

40	Entrée du nombre d'instructions ENV.
50-120	Dans ce sous-programme (lignes 430-530) sont entrés les paramètres pour les n1 instructions ENV. Les instructions sont exécutées directement (ligne 170) puis sorties ensuite dans la première fenêtre (lignes 80-110). Comme les entrées de paramètres sont identiques aux entrées pour l'instruction ENT, on emploie ici la technique du sous-programme. Les entrées interdites ne sont pas interceptées. Pour sortir toutes les instructions (c'est-à-dire les instructions ENT+ENV+SOUND), on dispose au total dans la partie droite de l'écran de 25 lignes. Si cette place est insuffisante, un scrolling de l'écran (glissement vers le haut) est inévitable.
130-210	Les instructions ENT sont déterminées suivant le même modèle (voyez les lignes 40-120).
220-420	Les instructions SOUND sont fixées et exécutées suivant le modèle du second programme du chapitre 7.3. Ici sont cependant pris en compte tous les paramètres de cette instruction. Après l'exécution, vous avez la possibilité de répéter cette opération ou de mettre fin au programme (ligne 400-420).
430-530	Sous-programme d'entrée des paramètres de courbe d'enveloppe.

7.5. Autres exemples

Vous pouvez utiliser les quatre exemples de programmes suivants pour l'accompagnement musical de vos programmes. Nous laissons naturellement à votre imagination le soin de juger si les titres choisis l'ont toujours été à bon escient. Ici également vous pourrez constater que de petites modifications peuvent entraîner des effets importants. Essayez vous-même !

```
10 REM Melodie aleatoire
20 SOUND 1,RND*4095
30 SOUND 2,RND*3500
40 SOUND 4,RND*2500
50 GOTO 20
```

```
10 REM Composition
20 REM ***********
30 ENT -1,3,1,1,3,-1,1:REM notes en legere vibration, la courbe
d'enveloppe se repete sans cesse pour la duree d'une note
40 DATA 236,25,213,25,190,50,236,25,213,25,190,50,236, 25,213,25
50 DATA 190,50,142,25,127,25,119,50,127,25,142,25,159,150
60 DATA 179,25,159,25,142,50,159,25,179,25,190,150,-1,0
70 READ t,d
80 IF t=-1 THEN RESTORE:GOTO 70:REM Arret avec 2*ESC
90 SOUND 2,t,d,12,0,1
100 SOUND 1,t/2,d,12,0,1
110 SOUND 4,t*2,d,12,0,1
120 GOTO 70
```

```
10 REM Soucoupe volante
20 FOR t=200 TO 300 STEP 10
30 SOUND 1,t,1
40 NEXT t
50 FOR t=300 TO 200 STEP -10
60 SOUND 1,t,1
70 NEXT t
80 GOTO 20
10 REM Alarme
20 FOR n=600 TO 160 STEP -20
30 SOUND 1,n,3
40 NEXT n
60 GOTO 20
```

8.Le lecteur de disquette

8.1. Introduction

Nous n'avons pas abordé jusqu'ici l'emploi du lecteur de disquette. Vous savez bien sûr que les informations qui se trouvent dans la mémoire de travail de votre ordinateur sont perdues dès que vous éteignez l'ordinateur. C'est pourquoi vous stockez vos programmes avec SAVE sur une disquette pour pouvoir les charger à nouveau avec LOAD en cas de besoin. Vous connaissez peut-être également déjà les instructions CHAIN, CHAIN MERGE et MERGE qui vous permettent de mélanger différents programmes. Cette possibilité est particulièrement importante lorsque la longueur de vos programmes excède la capacité mémoire de votre ordinateur. Mais pour le travail de programmation pratique il peut aussi être intéressant de pouvoir intégrer par exemple des parties de programme venant d'un programme 'étranger'.

Le lecteur de disquette permet d'autre part de stocker des masses de données importantes. Il arrive dans de nombreux programmes que des informations (des données) soient écrites, pendant le déroulement du programme, sur la disquette ou soient au contraire chargées à partir de la disquette (utilisation interactive de la disquette). L'ensemble des données forme alors ce qu'on appelle un fichier. Le présent chapitre essaiera donc de vous montrer comment on peut travailler sur de tels fichiers avec le BASIC AMSTRAD et le bloc de 64 K supplémentaires de votre CPC 6128.

8.2. Stockage séquentiel des données

Le stockage séquentiel des données signifie tout simplement que les données sont stockées caractère par caractère, à la suite les unes des autres. Les données ne peuvent être lues que dans l'ordre dans lequel elles ont été stockées. Si vous voulez par exemple lire le dernier caractère d'un fichier, vous devez

obligatoirement lire d'abord la totalité du fichier, c'est-à-dire passer sur tous les caractères précédents. Ce mode de traitement des données est aussi appliqué lorsqu'on utilise un lecteur de cassette.

Dans ce cas toutefois le chargement et la sauvegarde se déroulent beaucoup plus lentement que sur disquette.

Pour les problèmes peu complexes, le stockage séquentiel des données est tout à fait suffisant. Nous expliquerons son principe un peu plus bas en nous appuyant sur un exemple.

De nombreux ouvrages techniques consacrés au traitement électronique des données prennent la réalisation d'un programme de gestion d'un fichier d'adresses comme exemple d'emploi interactif de la disquette. La réalisation d'un programme d'agenda téléphonique constitue également un type d'application volontiers présenté (voyez le manuel d'utilisation). Pour vous présenter ici une autre domaine d'application, nous avons choisi pour ce chapitre la réalisation et le traitement d'un fichier de livres. Ce programme vous permettra de gérer votre stock de livres.

Avant de s'asseoir devant l'ordinateur, il convient de se demander d'abord ce qu'il va s'agir de stocker. Dans notre exemple, il serait intéressant de stocker pour chaque livre les informations suivantes :

- Nom de l'auteur
- Prénom de l'auteur
- Titre de l'ouvrage

Ces trois informations sur un livre formeront un ensemble qu'on appelle enregistrement (record en anglais). L'ensemble du fichier se compose donc d'enregistrements, chaque enregistrement contenant les informations sur un livre. Nos enregistrements se composent des trois informations citées plus haut. Ces différentes informations sont appelées également champs de données ou simplement champs. Notre premier enregistrement pourrait se présenter ainsi :

- Champ 1 : Kowal
- Champ 2 : Bernd
- Champ 3 : Des idées pour le CPC

Les instructions PRINT et WRITE vous permettent d'écrire les données sur la disquette.

Il faut indiquer pour cela le numéro de canal 9. Il vous faut cependant, auparavant, ouvrir un fichier de sortie avec l'instruction OPENOUT. Ce fichier devra être ultérieurement refermé avec l'instruction CLOSEOUT. Pour charger des données, vous utilisez les instructions OPENIN, INPUT ou LINE INPUT (avec le numéro de canal 9) et CLOSEIN. Naturellement, lors de l'entrée des données, vous devez tenir compte du type des données qui avaient été stockées ainsi que de la manière dont elles avaient été stockées. Au chapitre 8.2.1, nous essaierons d'éclaircir d'une part les différences entre PRINT et WRITE ainsi qu'entre INPUT et LINE INPUT d'autre part.

Le nombre de champs de données qui doivent être lus est également important. Si vous le connaissez déjà, vous pouvez organiser la routine de lecture des données en conséquence. Si ce n'est pas le cas, vous devez examiner avec la fonction EOF si la fin du fichier a été atteinte. Comme, dans l'exemple de programme que nous vous donnons plus loin, nous ne travaillons pas avec la fonction EOF, voici un petit exemple vous montrant comment organiser une routine flexible de chargement des données :

```
10 REM Routine de lecture flexible
20 OPENIN"Fichier"
30 WHILE NOT EOF
40 INPUT #9,a$
50 PRINT a$
60 WEND
70 CLOSEIN
80 END
```

Comme les différents champs de données sont ici chargés un à un et qu'on teste ensuite chaque fois si l'on a rencontré EOF (la fin du fichier), une erreur EOF Met est exclue avec ce programme.

Revenons à notre exemple. Bien entendu, nous ne voulons pas seulement stocker puis charger les noms des auteurs et les titres des livres, nous voulons également pouvoir exploiter ou traiter ces données.

C'est pourquoi, outre les fonctions de base du programme telles que l'entrée des données (au clavier) et la sortie sur écran, vous disposez également des fonctions suivantes :

- Recherche de données
- Correction de données
- Suppression de données

La fonction 'recherche de données' peut notamment se révéler très importante. C'est ainsi que l'ordinateur sortira par exemple, après que vous ayez entré un mot, le titre du livre dans lequel ce mot se présente. Si vous travaillez souvent de manière 'scientifique', vous pouvez aussi constituer des champs supplémentaires contenant des indications succinctes sur le contenu de chaque livre. Si vous voulez alors réunir la littérature sur un sujet déterminé, vous n'aurez qu'à demander à votre ordinateur de faire les recherches nécessaires. Le stockage de citations caractéristiques pourrait également rendre de grands services. Vous voyez donc qu'il est tout à fait recommandé de compléter ce programme en fonction de vos besoins personnels.

- **Mais il est temps que nous en venions au listing du programme :**

```
10 REM Fichier de livres
20 DIM n$(500),v$(500),t$(500)
30 REM Menu
40 MODE 1:PRINT TAB(32)"Entree":PRINT:PRINT
50 PRINT"Entree de donnees";TAB(35)"1":PRINT
```

```
60 PRINT"Sortie ecran";TAB(35)"2":PRINT
70 PRINT"Stocker donnees";TAB(35)"3":PRINT
80 PRINT"Charger donnees";TAB(35)"4":PRINT
90 PRINT"Recherche de donnees";TAB(35)"5":PRINT
100 PRINT"Correction de donnees";TAB(35)"6":PRINT
110 PRINT"Suppression de donnees";TAB(35)"7":PRINT
120 PRINT"Fin du programme";TAB(35)"8":PRINT
130 PRINT:PRINT:PRINT:INPUT"Votre choix ";a
140 IF a<1 OR a>8 THEN 40
150 MODE 2:ON a GOTO 160,260,400,480,560,870,970,1110
160 WINDOW #1,1,80,23,25:WINDOW #0,1,80,1,20
170 PRINT#1,"Les entrees se terminent si vous appuyez simplement
sur la touche"
180 PRINT#1,"RETURN pour 'Nom de l'auteur' !"
190 k=k+1
200 PRINT "Enregistrement"k;":":PRINT:PRINT
210 GOSUB 1160:LINE INPUT n$(k):PRINT
220 IF n$(k)="" THEN k=k-1:WINDOW #0,1,80,1,25:GOTO 40
230 GOSUB 1170:LINE INPUT v$(k):PRINT
240 GOSUB 1180:LINE INPUT t$(k):PRINT:PRINT:PRINT
250 GOTO 190
260 PRINT"Il y a";k;"enregistrements
disponibles":PRINT:PRINT:PRINT
270 IF k<1 THEN GOSUB 1190:GOTO 40
280 PRINT"Sortie":PRINT:PRINT
290 INPUT"du numero d'enreg. ";aa:PRINT
300 IF aa<1 OR aa>k THEN 290
310 INPUT"au numero d'enreg. ";ae
320 IF ae<1 OR ae>k THEN 310
330 CLS:FOR i=aa TO ae
340 PRINT "Enregistrement"k":":PRINT:PRINT:PRINT
350 GOSUB 1160:PRINT": ";n$(i):PRINT
360 GOSUB 1170:PRINT": ";v$(i):PRINT
370 GOSUB 1180:PRINT": ";t$(i)
380 GOSUB 1190
390 NEXT i:GOTO 40
400 OPENOUT"Fichier"
410 PRINT#9,k
420 FOR i=1 TO k
430 PRINT#9,n$(i)
440 PRINT#9,v$(i)
```

```
450 PRINT#9,t$(i)
460 NEXT i
470 CLOSEOUT:GOTO 40
480 OPENIN"Fichier"
490 INPUT#9,k
500 FOR i=1 TO k
510 LINE INPUT#9,n$(i)
520 LINE INPUT#9,v$(i)
530 LINE INPUT#9,t$(i)
540 NEXT i
550 CLOSEIN:GOTO 40
560 PRINT"Que faut-il rechercher ?":PRINT:PRINT:PRINT
570 PRINT TAB(50)"Entree":PRINT:PRINT
580 GOSUB 1160:PRINT TAB(53)"1":PRINT
590 GOSUB 1170:PRINT TAB(53)"2":PRINT
600 GOSUB 1180:PRINT TAB(53)"3":PRINT
610 PRINT"Theme (dans un titre de livre)";:PRINT
TAB(53)"4":PRINT
620 PRINT:PRINT:INPUT"Votre choix ";a
630 IF a<1 OR a>4 THEN CLS:GOTO 560
640 PRINT:PRINT:PRINT:INPUT"Entrez le texte recherche ";s$:CLS
650 ON a GOTO 660,690,720,750
660 FOR i=1 TO k
670 IF s$=n$(i) THEN GOSUB 790
680 NEXT i:GOTO 780
690 FOR i=1 TO k
700 IF s$=v$(i) THEN GOSUB 790
710 NEXT i:GOTO 780
720 FOR i=1 TO k
730 IF s$=t$(i) THEN GOSUB 790
740 NEXT i:GOTO 780
750 FOR i=1 TO k
760 IF INSTR(t$(i),s$)>0 THEN GOSUB 790
770 NEXT i
780 PRINT:PRINT:PRINT"Texte recherche non trouve ou trouve dans
aucun autre enreg.":GOSUB 1190:GOTO 40
790 PRINT"Le texte recherche se trouve dans
l'enregistrement";i;":":PRINT:PRINT:PRINT
800 GOSUB 1160:PRINT n$(i):PRINT
810 GOSUB 1170:PRINT v$(i):PRINT
820 GOSUB 1180:PRINT t$(i):PRINT:PRINT:PRINT
```

```
830 INPUT"Faut-il continuer a chercher (o,n) ";a$
840 IF a$="n" THEN 40
850 IF a$<>"o" THEN 830
860 CLS:RETURN
870 INPUT"Quel enregistrement faut-il corriger ";dk:PRINT:PRINT
880 IF dk<1 OR dk>k THEN 870
890 GOSUB 1160:PRINT n$(dk):PRINT
900 GOSUB 1170:PRINT v$(dk):PRINT
910 GOSUB 1180:PRINT t$(dk):PRINT:PRINT
920 PRINT"Rectification:":PRINT:PRINT:PRINT
930 GOSUB 1160:LINE INPUT n$(dk):PRINT
940 GOSUB 1170:LINE INPUT v$(dk):PRINT
950 GOSUB 1180:LINE INPUT t$(dk):PRINT:PRINT:PRINT
960 GOTO 40
970 INPUT"Quel enregistrement faut-il supprimer ";dk:PRINT:PRINT
980 IF dk<1 OR dk>k THEN 970
990 GOSUB 1160:PRINT n$(dk):PRINT
1000 GOSUB 1170:PRINT v$(dk):PRINT
1010 GOSUB 1180:PRINT t$(dk):PRINT:PRINT:PRINT
1020 INPUT"supprimer vraiment (o,n) ";a$
1030 IF a$="n" THEN 40
1040 IF a$<>"o" THEN 1020
1050 FOR i=dk TO k-1
1060 n$(i)=n$(i+1)
1070 v$(i)=v$(i+1)
1080 t$(i)=t$(i+1)
1090 NEXT i
1100 k=k-1:GOTO 40
1110 MODE 1:PRINT"Avez-vous bien tout stocke ?"
1120 PRINT:PRINT"Sinon, entrez simplement GOTO 40."
1130 PRINT:PRINT:PRINT"Au revoir !":PRINT:PRINT
1140 END
1150 REM Sous-programme pour le texte d'entree-sortie
1160 PRINT"Nom de l'auteur    ";:RETURN
1170 PRINT"Prenom de l'auteur ";:RETURN
1180 PRINT"Titre du livre     ";:RETURN
1190 REM SP Attendre
1200 LOCATE 5,25
1210 PRINT"Frappez une touche S.V.P. !"
1220 a$=INKEY$:IF a$="" THEN 1220
1230 CLS:RETURN
```

❐ Liste de variables :

a	Réponse 'choix du menu'
a$	Chaîne de réponse (o,n)
aa	Premier enregistrement dans la partie 'sortie écran' du programme
ae	Dernier enregistrement dans la partie 'sortie écran' du programme.
dk	Numéro de l'enregistrement à corriger.
i	Index de comptage.
k	Nombre d'enregistrements.
n$(i)	Nom de l'auteur
s$	Texte recherché
t$(i)	Titre du livre
v$(i)	Prénom de l'auteur

❐ Description du programme :

10	Commentaire.
20	Réservation de place mémoire pour chaque fois 500 noms, prénoms et titres.
30-120	Commentaire, passage en mode 40 colonnes et sortie du menu.
130-150	Entrée d'une valeur pour la sélection du menu, passage en mode 80 colonnes et sauts de programme.
160-250	Entrée des enregistrements. La ligne 160 définit deux fenêtres écran à cet effet. Alors qu'un commentaire sur l'interruption de l'entrée sera sorti dans la fenêtre 1, la fenêtre 0 sert à l'entrée des différents champs de données. La variable k est augmentée de 1 avant chaque entrée. k sert donc ainsi d'index pour les 3 champs de l'enregistrement. Pour pouvoir stocker également les virgules et les guillemets, on travaille ici avec LINE INPUT.

	Avant chaque entrée, un texte sorti à l'aide des sous -programmes (lignes 1160-1180). En ligne 250 est effectué un retour à la poursuite de l'entrée. En cas d'interruption de l'entrée (ligne 220), la variable k qui avait déjà été augmentée de 1 est à nouveau diminuée, la fenêtre 0 est à nouveau définie (écran entier) et on retourne au menu principal (à partir de la ligne 40).
260-390	Sortie écran des enregistrements. Après une sortie de commentaire (ligne 260) une entrée permet à l'utilisateur d'indiquer les enregistrements qui doivent être sortis. Les entrées incorrectes ou impossibles sont naturellement interceptées. Les lignes 330-390 sortent alors les enregistrements avec chaque fois une interruption provoquée par la boucle d'attente (sous-programme 'attendre', ligne 1190). Un retour au menu principal est ensuite effectué.
400-470	Sortie des données sur disquette et retour au menu principal. Avant que les données ne soient sauvegardées enregistrement par enregistrement, la valeur de la variable est transférée sur la disquette (voyez la routine de chargement des données).
480-550	Les données sont ici lues exactement comme elles avaient été stockées. La valeur k est lue en premier pour pouvoir former ensuite une boucle de lecture des enregistrements. On travaille ici avec LINE INPUT pour pouvoir charger également les virgules et les guillemets (voyez à ce sujet également le chapitre 8.2.1).

560-860	Cette partie du programme recherche un texte entré par l'utilisateur. Il peut s'agir d'un nom ou d'un prénom, d'un titre ou d'un terme. Après un choix dans un menu (lignes 560-630) et l'entrée du texte recherché (ligne 640), l'opération de recherche proprement dite se déroule dans les lignes 660-770. Les lignes 660-680 recherchent un nom, les lignes 690-710 un prénom, les lignes 720-740 un titre et les lignes 750-770 un terme (le saut approprié a été effectué en ligne 650 en fonction du choix dans le menu). Chaque fois que le texte recherché a été trouvé, on saute au sous-programme en ligne 790 pour sortir l'enregistrement correspondant. Suivant le choix de l'utilisateur du programme (voyez la ligne 830), la recherche peut se poursuivre ou bien on retourne au menu principal. Une fois le travail de recherche terminé, c'est-à-dire une fois que tous les enregistrements ont été examinés, un sortie de commentaire est effectuée en ligne 780 et on retourne au menu principal.
870-960	Correction d'un enregistrement. L'enregistrement voulu est tout d'abord sorti. Vous pouvez alors redéfinir les champs correspondants puis on retourne au menu.
970-1100	Cette partie du programme permet de supprimer un enregistrement. Après que l'enregistrement et une question de sécurité aient été sortis (ligne 1020), l'enregistrement est supprimé dans les lignes 1050-1090. Si la mémoire programme contient par exemple 20 enregistrements et s'il faut supprimer le 10ième enregistrement, alors le 11ème enregistrement deviendra le 10ème, le 12ème deviendra le 11ème et le 20ème le 19ème. Comme le dernier enregistrement existe maintenant en deux exemplaires (en 19ème et en 20ème positions), k est diminué de 1 avant le retour au menu principal.
1110-1140	Le programme se termine après une sortie de commentaire.

1150-1180	Sous-programmes pour les textes souvent utilisés.
1190-1230	Sous-programme "attendre" et vidage de l'écran.

8.2.1. Comparaison de PRINT et WRITE, INPUT et LINE INPUT

Aussi étonnant que cela puisse paraître, l'entrée-sortie sur la disquette ne se distingue pas outre mesure de l'entrée-sortie sur l'écran. Examinons un peu l'exemple suivant :

```
10 REM Exemple 1
20 a$="a":b$="b":c$="c"
30 OPENOUT"Fichier"
40 PRINT#9,a$,b$,c$
50 CLOSEOUT
60 OPENIN"Fichier"
70 INPUT#9,q$
80 PRINT q$
90 CLOSEIN
```

Contrairement à la routine de sortie du programme du dernier chapitre, les différents champs ou variables sont ici énumérés dans une instruction PRINT séparés uniquement par un point-virgule. Cela entraîne que, un peu comme pour la sortie écran, les différentes variables seront écrites sur la disquette les unes à la suite des autres. La sortie sur disquette se distingue toutefois en ceci que ces données écrites les unes à la suite des autres constituent un seul champ de données, ce qui signifie qu'un champ ou une variable suffira pour recevoir les trois valeurs de variables. C'est pourquoi vous obtiendrez :

```
abc
```

pour q$. Si vous utilisez en ligne 40 des virgules à la place des points-virgules, vous obtiendrez le résultat suivant :

```
a        b        c
```

Comme pour la sortie écran, des espaces ont donc été placés sur la disquette (une méthode de programmation qui n'est pas très économe de place de stockage). Cependant une variable suffit toujours à recevoir les trois valeurs.

Vous l'avez sans doute déjà deviné : il faut placer entre les données la marque de séparation logique qu'est le Carriage Return (retour de chariot, touche RETURN ou ENTER) pour que les différents champs soient aussi reconnus sur la disquette comme étant indépendants. Si vous formez donc deux lignes supplémentaires pour stocker b$ et c$ dans une instruction PRINT séparée, vous n'obtiendrez plus que la valeur 'a' pour q$.

Ce que nous avons dit jusqu'ici ne s'applique toutefois qu'aux variables de texte. Pour les variables numériques, vous pouvez tout à fait placer plusieurs variables dans une seule instruction PRINT et celles-ci seront aussi reconnues comme indépendantes lors de leur lecture.

Ajoutons encore au sujet de l'instruction INPUT qu'il est indifférent que vous lisiez dans une instruction INPUT un enregistrement entier ou que vous lisiez un enregistrement champ par champ.

Mais que se passera-t-il si la ligne 40 écrit le texte "oui,oui" sur la disquette ? Vous n'obtiendrez pas alors :

```
oui,oui
```

pour q$, comme vous auriez pu vous y attendre, mais seulement :

```
oui
```

La virgule est donc comprise par l'instruction INPUT comme une marque de séparation.

Lorsque des champs de données sont lus avec INPUT, les caractères suivants sont considérés comme marques de séparation :

- Carriage Return
- Virgule
- Marque EOF

Rappelons ici que la virgule et le retour de chariot fonctionnent également comme marque de séparation pour l'entrée au clavier.

Pour la lecture de variables numériques, il convient de rajouter encore l'espace à la liste des symboles de séparation.

Lorsque vous utilisez l'instruction WRITE, la chaîne de caractères est placée entre guillemets. Lors de la lecture de cette chaîne, toutes les virgules seront alors acceptées. Les guillemets ne seront cependant pas lus comme faisant partie de la chaîne (ce qui n'est d'ailleurs pas généralement le but recherché).

Pour chaque application, il convient de se demander quelles instructions il est préférable d'employer. On ne peut absolument pas dire que telle ou telle instruction soit supérieure à une autre. Cela dépend toujours de la situation considérée.

Examinons, à ce propos, un second exemple qui reflète assez précisément la situation de départ dans notre programme de gestion d'un fichier de livres :

```
10 REM Exemple 2
20 LINE INPUT a$
30 OPENOUT"Fichier"
40 WRITE#9,a$
50 CLOSEOUT
60 OPENIN"Fichier"
70 INPUT#9,q$
80 PRINT q$
90 CLOSEIN
```

Nous ne nous intéressons bien sûr plus maintenant qu'au cas particulier où une chaîne est entrée en ligne 20 avec virgules et guillemets. Cela entraînera dans le cas présent que si les

virgules seront bien acceptées, les guillemets seront par contre considérés comme des marques de séparation.

Si nous modifions le programme en utilisant l'instruction PRINT à la ligne 40, les guillemets seront dans ce cas bien acceptés comme il convient mais les virgules seront comprises comme marques de séparation.

Mais une autre modification du programme nous apportera la solution de ce problème. Nous laisserons l'instruction PRINT en ligne 40 mais en ligne 70 nous ne lirons pas la chaîne avec INPUT mais avec LINE INPUT. C'est d'ailleurs exactement ainsi que nous avons procédé dans notre programme de gestion d'un fichier de livres.

Si vous aviez pensé à utiliser l'instruction WRITE en ligne 40, nous vous conseillons d'essayer tout simplement par vous-même de voir ce qui se passe dans ce cas.

8.3. Les fichiers relatifs avec AMSDOS

La programmation de fichiers relatifs présente un certain nombre d'avantages par rapport à la gestion séquentielle de fichiers. La particularité caractéristique des fichiers relatifs réside dans le libre accès aux différents enregistrements (d'où également les notions 'accès sélectif' ou 'random access'). Si vous avez par exemple un fichier de 100 enregistrements et que vous voulez lire le 66ème, vous serez obligé, avec un fichier séquentiel, de lire tous les enregistrements précédents pour parvenir au 66ème enregistrement qui vous intéresse. Avec un fichier relatif, par contre, vous pouvez lire directement l'enregistrement voulu. La recherche d'enregistrements déterminés, avec des boucles plus ou moins complexes, telle qu'elle est d'usage avec les fichiers séquentiels, devient donc également superflue. Il faut par contre respecter avec les fichiers relatifs un certain nombre de règles que nous expliquerons un peu plus loin. Un autre aspect est le fait qu'il ne soit plus nécessaire de charger un fichier entièrement en mémoire pour pouvoir le traiter. On n'a donc plus à se préoccuper de la place mémoire consommée par autant

d'instructions DIM lorsqu'il s'agit de charger entièrement en mémoire un fichier de taille importante.

La gestion relative de fichiers présente cependant également quelques inconvénients de sorte que son emploi n'est réellement intéressant que lorsque des accès au fichier sont souvent nécessaires. Pour la plupart des applications réduites on peut donc parfaitement en rester à la programmation séquentielle de fichiers.

Mais quels sont donc les inconvénients, ou plus précisément les contraintes, dont il faut tenir compte ?

Commençons d'abord par les réflexions qui sont nécessaires au départ. Dans un fichier séquentiel, les enregistrements peuvent avoir n'importe quelle taille. Dans un fichier relatif, par contre, il faut bien réfléchir, avant de commencer, à la longueur maximale que pourra atteindre un enregistrement. Par longueur d'un enregistrement, nous entendons ici le nombre de caractères par enregistrement.

Si un enregistrement comporte plusieurs champs, ce qui sera généralement le cas, il faut également fixer dès le départ la longueur maximale des différents champs de données. Pour notre programme de gestion de fichier de livres du chapitre précédent, nous pourrions attribuer aux trois champs de données les longueurs maximales suivantes :

- Champ 1 : 30 caractères
- Champ 2 : 20 caractères
- Champ 3 : 100 caractères

Un nom d'auteur ne devra donc pas comporter plus de 30 caractères, le prénom est limité à 20 caractères et le titre du livre peut comprendre jusqu'à 100 caractères. Nous arrivons ainsi à une longueur d'enregistrement de 150 caractères.

Disons tout de suite qu'il n'est pas normalement nécessaire de faire attention aux longueurs prescrites lors de l'entrée des données au clavier. Les champs de données sont en effet

remplis avec des espaces lorsque les données entrées sont trop courtes et lorsqu'elles sont trop longues on ne prend en compte que les 20, 30 ou 100 premiers caractères. Il n'empêche que nous pouvons affirmer dès maintenant que la gestion relative de fichiers nécessite une connaissance plus précise de la structure des données.

Venons-en au problème principal que pose la programmation de fichiers relatifs. Comme vous le savez peut-être déjà, cela n'est pas possible avec le lecteur de disquette intégré. Seules des routines machine très complexes permettent de simuler des fichiers relatifs avec des fichiers séquentiels. Nous ne nous étendrons donc pas davantage sur cette possibilité d'autant que c'est tout à fait inutile puisque vous pouvez utiliser à cet effet le second bloc de 64 K de RAM.

Comme vous vous en souvenez peut-être, nous avons utilisé cette zone mémoire supplémentaire pour des opérations écran au chapitre 7 du présent ouvrage. Nous allons l'utiliser cette fois comme disque RAM, c'est-à-dire comme zone mémoire supplémentaire pour des blocs de texte.

Mais comme le BASIC n'utilise normalement que le premier bloc de 64 K, il faut à nouveau lançer le programme "BANK MANAGER" pour obtenir les instructions RSX supplémentaires qui permettent d'utiliser le second bloc de 64 K. Ces instructions peuvent alors être utilisées dans des programmes BASIC.

Et voici comment la gestion de fichier relatif se présente alors :

L'instruction : **|BANKOPEN,x**

vous permet d'ouvrir un fichier relatif de longueur d'enregistrement x. Dans l'exemple donné plus haut, nous choisirions donc pour x la valeur 150.

Avec : **|BANKWRITE,@r%,ds$,n**

vous pouvez écrire la chaîne ds$ dans le fichier en RAM. Il faut bien sûr veiller à ce que la longueur de la chaîne ds$ soit

conforme à la valeur indiquée pour x. Le paramètre n indique l'enregistrement dans lequel il s'agit d'écrire. Ce paramètre peut être omis, auquel cas on écrira toujours dans l'enregistrement actuel. La variable entière r% indique à tout moment le numéro de l'enregistrement actuel. Si une opération (écriture ou lecture) s'est déroulée avec succès, r% est automatiquement augmenté de 1. Si une erreur est apparue pour une raison ou pour une autre, un code d'erreur négatif sera sorti pour r%.

La variable r% doit être initialisée avant l'instruction d'écriture.

A cet effet, entrez : r%=0

pour faire commencer r% à 0.

L'instruction : |BANKREAD,@r%,in$,n

permet de charger un enregistrement du fichier RAM dans la variable de texte in$.

Il est important que in$ ait été auparavant initialisé comme une chaîne de caractères ayant la longueur qui convient (par exemple in$=SPACE$(150)). Le paramètre n peut être omis comme pour BANKWRITE.

L'instruction : |BANKFIND,@r%,s$,d,f

est très pratique pour une gestion de fichier relatif car elle permet de faire rechercher une chaîne s$ déterminée dans tous les enregistrements. Il faut veiller, ici encore, à ce que le texte recherché s$ ait bien la longueur d'enregistrement prescrite. Une opération de recherche fructueuse aura pour effet de faire de l'enregistrement dans lequel le texte recherché a été trouvé l'enregistrement actuel. Vous pouvez donc appeler le résultat de la recherche avec PRINT r%. Ce n'est que si la chaîne n'a pas été trouvée qu'un code négatif sera sorti pour r%. Les paramètres d et f permettent d'indiquer les numéros de l'enregistrement par lequel doit commencer la recherche et de celui avec lequel elle doit se terminer. Si on omet ces paramètres, la recherche concernera la totalité du fichier. Une

possibilité très intéressante pour la recherche est de placer avec CHR$(0) des caractères universels (Jokers ou Wild cards) dans le texte recherché. CHR$(0) pourra donc être mis pour un caractère quelconque. Si vous voulez par exemple rechercher un mot dans le troisième champ de données, vous pouvez, avant d'entrer le mot recherché pour le troisième champ de données, remplir le texte recherché de caractères universels. Par rapport à notre exemple de fichier de livres, cela voudrait dire qu'il faudrait remplir le texte recherché de 50 CHR$(0) avant d'entrer le mot recherché pour le titre. Nous aurions donc :

```
s$=STRING$(50,0)+mot pour le titre du livre
```

Une autre possibilité d'utilisation de l'instruction de recherche consiste à ne placer de joker que pour certaines lettres déterminées. Si vous ne savez plus par exemple si vous avez écrit 'Disk' ou 'Disc', il vous suffit alors d'écrire pour le texte ou le mot recherché :

```
s$="Dis"+CHR$(0)
```

Du fait de l'emploi du joker, il sera maintenant indifférent pour l'opération de recherche que vous ayez écrit ce mot avec 'c' ou avec 'k'.

Vous trouverez plus bas un programme qui utilise le disque RAM pour la gestion d'un fichier relatif. Nous avons pour cela repris le programme de gestion de fichier de livre du chapitre précédent, en y apportant les modifications nécessaires. Toutefois, comme nous le disions plus haut, les fichiers relatifs n'ont d'intérêt réel que lorsque des accès fréquents à la disquette sont nécessaires. Ce programme répond donc plus à un souci de démonstration qu'à la nécessité imposée par une application qui ne pourrait pour ainsi dire pas se passer des fichiers relatifs.

Disons donc d'emblée que le fichier de livres ne fonctionne pas vraiment mieux et qu'il n'est pas plus pratique que la version séquentielle. Comme le programme de gestion de fichier de livres n'a pas besoin d'accès fréquents à la disquette (le fichier se trouve toujours entièrement en mémoire), l'utilisateur ne

constatera pratiquement pas de différence dans l'emploi du programme. La version relative est même moins pratique puisqu'il faut absolument lancer le programme BANKMAN avant de lancer le programme. Sinon les instructions RSX supplémentaires ne seraient pas comprises ce qui entraînerait des messages d'erreur en conséquence.

A part la limitation de la longueur des champs de données, la seule différence pour l'utilisateur du programme réside dans la fonction 'Recherche de données' du programme. Alors que la version séquentielle vous offrait la possibilité extrêmement intéressante de faire rechercher un terme quelconque dans une position quelconque d'un titre de livre, vous devez ici remplir obligatoirement la totalité du champ de données lorsque vous entrez un texte à rechercher. Vous n'avez donc plus la possibilité de faire rechercher un terme apparaissant en n'importe quelle position d'un titre d'ouvrage. Vous devez savoir exactement en quelle position du champ de données ce terme figure. On ne pourrait remédier à cet inconvénient que par une programmation quelque peu compliquée (le lecteur intéressé peut considérer cela comme une suggestion pour une initiative personnelle).

La version relative de la recherche a cependant également l'avantage de permettre l'entrée de caractères joker. Vous pouvez donc fixer un texte à rechercher pour un des trois champs de données suivant le modèle de l'exemple Dis(k)(c) donné plus haut.

Nous laisserons au lecteur le soin de décider laquelle des deux versions du programme est la plus pratique pour lui. Mais quoi qu'il en soit, nous vous recommandons en tout cas d'examiner la version relative. Si vous avez déjà bien compris le fonctionnement de la version séquentielle, vous ne devriez pas avoir de mal à comprendre comment nous avons utilisé le disque RAM pour la gestion d'un fichier relatif.

Nous dirons cependant encore quelques mots sur la technique de programmation utilisée. Dans ce programme, toutes les données entrées ou chargées séquentiellement sont écrites directement dans le disque RAM. Il est donc superflu de

procéder à un dimensionnement de variables. La fonction 'stocker données' concerne ici la sauvegarde séquentielle des données sur la disquette. Les données sont à cet effet lues à partir du disque RAM puis stockées séquentiellement enregistrement après enregistrement. Ce qui est 'relatif' dans ce programme c'est que les opérations de sortie, recherche, correction et suppression s'effectuent en RAM et qu'on accède donc directement et 'librement' aux différents enregistrements. La différence par rapport à la version de ce programme donnée au chapitre précédent apparaîtrait certainement plus nettement si nous n'avions pas créé, dans cette première version, les tableaux n$(i), v$(i) et t$(i) pour les champs de données et si nous avions stocké et lu sur la disquette chaque champ de données isolément. Dans ce cas en effet, il nous aurait fallu un accès à la disquette (à un fichier séquentiel) pour chaque opération. Un tel accès à la disquette entraîne une perte de temps sans comparaison avec les délais très courts nécessaires pour les 'libres' accès au disque RAM. Peut-être penserez-vous toutefois que, si les fichiers relatifs sont surtout intéressants lorsqu'il s'agit de traiter des fichiers de taille importante, 64 K ne représentent pas une masse très élevée de données. En cela vous avez tout à fait raison. Pour notre programme de fichier de livres, la limite est d'environ 400 enregistrements.

Seul l'emploi d'une astuce permettrait de gérer un nombre plus important d'enregistrements. Cette astuce consiste à gérer des fichiers plus longs en les promenant entre le lecteur de disquette et le disque RAM. Une possibilité certainement très intéressante pour un programmeur confirmé.

- **Mais venons-en enfin au listing du programme :**

```
10 REM Fichier de livres avec disque RAM
20 |BANKOPEN,150:r%=0
30 ds$=SPACE$(150)
40 REM Menu
50 MODE 1:PRINT TAB(32)"Entree":PRINT:PRINT
60 PRINT"Entrer donnees";TAB(35)"1":PRINT
70 PRINT"Sortie ecran";TAB(35)"2":PRINT
80 PRINT"Stocker donnees";TAB(35)"3":PRINT
90 PRINT"Charger donnees";TAB(35)"4":PRINT
```

```
100 PRINT"Chercher donnees";TAB(35)"5":PRINT
110 PRINT"Corriger donnees";TAB(35)"6":PRINT
120 PRINT"Supprimer donnees";TAB(35)"7":PRINT
130 PRINT"Fin du programme";TAB(35)"8":PRINT
140 PRINT:PRINT:PRINT:INPUT"Votre choix ";a
150 IF a<1 OR a>8 THEN 50
160 MODE 2:ON a GOTO 170,250,400,470,540,970,1070,1210
170 WINDOW #1,1,80,23,25:WINDOW #0,1,80,1,20
180 PRINT#1,"L'entree se termine si vous appuyez seulement sur la touche"
190 PRINT#1,"RETURN pour 'Nom de l'auteur' !"
200 k=k+1
210 PRINT "Enregistrement"k":":PRINT:PRINT
220 GOSUB 1340
230 |BANKWRITE,@r%,n$+v$+t$,k
240 GOTO 200
250 PRINT"Il y a";k;"enregistrements
disponibles":PRINT:PRINT:PRINT
260 IF k<1 THEN GOSUB 1450:GOTO 50
270 PRINT"Sortie":PRINT:PRINT
280 INPUT"du numero d'enreg. ";aa:PRINT
290 IF aa<1 OR aa>k THEN 280
300 INPUT"au numero d'enreg. ";ae
310 IF ae<1 OR ae>k THEN 300
320 CLS:FOR i=aa TO ae
330 PRINT "Enregistrement"i":":PRINT:PRINT:PRINT
340 |BANKREAD,@r%,ds$,i
350 GOSUB 1260:GOSUB 1300:PRINT": ":PRINT:PRINT
n$:PRINT:PRINT
360 GOSUB 1270:GOSUB 1310:PRINT": ":PRINT:PRINT
v$:PRINT:PRINT
370 GOSUB 1280:GOSUB 1320:PRINT": ":PRINT:PRINT t$
380 GOSUB 1450
390 NEXT i:GOTO 50
400 OPENOUT"Fichier"
410 PRINT#9,k
420 FOR i=1 TO k
430 |BANKREAD,@r%,ds$,i
440 PRINT#9,ds$
450 NEXT i
460 CLOSEOUT:GOTO 50
470 OPENIN"Fichier"
```

```
480 INPUT#9,k
490 FOR i=1 TO k
500 LINE INPUT#9,ds$
510 |BANKWRITE,@r%,ds$,i
520 NEXT i
530 CLOSEIN:GOTO 50
540 PRINT"Dans quel champ faut-il chercher ?":PRINT:PRINT:PRINT
550 PRINT TAB(50)"Entree":PRINT:PRINT
560 GOSUB 1260:GOSUB 1300:PRINT TAB(53)"1":PRINT
570 GOSUB 1270:GOSUB 1310:PRINT TAB(53)"2":PRINT
580 GOSUB 1280:GOSUB 1320:PRINT TAB(53)"3":PRINT
590 PRINT:PRINT:INPUT"Votre choix ";a
600 CLS:IF a<1 OR a>3 THEN 540
610 PRINT"Entrez maintenant le texte a rechercher. Marquez avec
la touche"
620 PRINT"COPY l'emplacement des caracteres universels
(Joker).":PRINT:PRINT:PRINT
630 ka=1:s$=""
640 ON a GOTO 650,700,750
650 sl=30:GOSUB 810
660 s$=s$+STRING$(120,0)
670 |BANKFIND,@r%,s$,ka,k
680 IF r%<0 THEN 800
690 GOSUB 880:IF r%<k THEN ka=r%+1:GOTO 670 ELSE 800
700 sl=20:GOSUB 810
710 s$=STRING$(30,0)+s$+STRING$(100,0)
720 |BANKFIND,@r%,s$,ka,k
730 IF r%<0 THEN 800
740 GOSUB 880:IF r%<k THEN ka=r%+1:GOTO 720 ELSE 800
750 sl=100:GOSUB 810
760 s$=STRING$(50,0)+s$
770 |BANKFIND,@r%,s$,ka,k
780 IF r%<0 THEN 800
790 GOSUB 880:IF r%<k THEN ka=r%+1:GOTO 770
800 PRINT:PRINT:PRINT"Texte recherche non trouve ou trouve dans
aucun autre enregistrement.":GOSUB 1450:GOTO 50
810 FOR i=1 TO sl
820 x$=INKEY$:IF x$="" THEN 820
830 IF INKEY(9)=0 THEN x$=CHR$(0):PRINT"_";:GOTO 850
840 PRINT x$;
850 s$=s$+x$
```

```
860 NEXT i
870 RETURN
880 CLS:PRINT"Le texte recherche se trouve dans
l'enregistrement";r%;" :":PRINT:PRINT:PRINT
890 |BANKREAD,@r%,ds$,r%
900 GOSUB 1260:GOSUB 1300:PRINT:PRINT n$:PRINT:PRINT
910 GOSUB 1270:GOSUB 1310:PRINT:PRINT v$:PRINT:PRINT
920 GOSUB 1280:GOSUB 1320:PRINT:PRINT t$:PRINT:PRINT:PRINT
930 INPUT"Faut-il encore chercher (o,n) ";a$
940 IF a$="n" THEN 50
950 IF a$<>"o" THEN 930
960 CLS:RETURN
970 INPUT"Quel enregistrement faut-il corriger ";dk:PRINT:PRINT
980 IF dk<1 OR dk>k THEN 970
990 |BANKREAD,@r%,ds$,dk
1000 GOSUB 1260:GOSUB 1300:PRINT n$:PRINT
1010 GOSUB 1270:GOSUB 1310:PRINT v$:PRINT
1020 GOSUB 1280:GOSUB 1320:PRINT t$:PRINT:PRINT
1030 PRINT"Correction: ":PRINT:PRINT:PRINT
1040 GOSUB 1340
1050 |BANKWRITE,@r%,n$+v$+t$,dk
1060 GOTO 50
1070 INPUT"Quel enregistrement faut-il supprimer
";dk:PRINT:PRINT
1080 IF dk<1 OR dk>k THEN 1070
1090 |BANKREAD,@r%,ds$,dk
1100 GOSUB 1260:GOSUB 1300:PRINT n$:PRINT
1110 GOSUB 1270:GOSUB 1310:PRINT v$:PRINT
1120 GOSUB 1280:GOSUB 1320:PRINT t$:PRINT:PRINT:PRINT
1130 INPUT"supprimer vraiment (o,n) ";a$
1140 IF a$="n" THEN 50
1150 IF a$<>"o" THEN 1130
1160 FOR i=dk TO k-1
1170 |BANKREAD,@r%,ds$,dk+1
1180 |BANKWRITE,@r%,ds$,dk
1190 NEXT i
1200 k=k-1:GOTO 50
1210 MODE 1:PRINT"Tout est bien sauvegarde ?"
1220 PRINT:PRINT"Sinon entrez simplement GOTO 40."
1230 PRINT:PRINT:PRINT"Au revoir !":PRINT:PRINT
1240 END
```

```
1250 REM Sous-programmes pour determiner les champs de donnees
1260 n$=LEFT$(ds$,30):RETURN
1270 v$=MID$(ds$,31,20):RETURN
1280 t$=RIGHT$(ds$,100):RETURN
1290 REM Sous-programmes pour le texte d'entree-sortie
1300 PRINT"Nom de l'auteur    ";:RETURN
1310 PRINT"Prenom de l'auteur ";:RETURN
1320 PRINT"Titre du livre     ";:RETURN
1330 REM Sous-programme de definition des champs de donnees
1340 GOSUB 1300:LINE INPUT n$:PRINT
1350 IF n$="" THEN k=k-1:WINDOW #0,1,80,1,25:GOTO 50
1360 IF LEN(n$)<30 THEN n$=n$+SPACE$(30-LEN(n$))
1370 IF LEN(n$)>30 THEN n$=LEFT$(n$,30)
1380 GOSUB 1310:LINE INPUT v$:PRINT
1390 IF LEN(v$)<20 THEN v$=v$+SPACE$(20-LEN(v$))
1400 IF LEN(v$)>20 THEN v$=LEFT$(v$,20)
1410 GOSUB 1320:LINE INPUT t$:PRINT:PRINT:PRINT
1420 IF LEN(t$)<100 THEN t$=t$+SPACE$(100-LEN(t$))
1430 IF LEN(t$)>100 THEN t$=LEFT$(t$,100)
1440 RETURN
1450 REM SP Attendre
1460 LOCATE 5,25
1470 PRINT"Frappez une touche S.V.P. !"
1480 a$=INKEY$:IF a$="" THEN 1480
1490 CLS:RETURN
```

❐ Liste de variables :

a	Réponse 'choix du menu'
a$	Chaîne de réponse (o,n)
aa	Premier enregistrement dans la partie 'sortie écran' du programme
ae	Dernier enregistrement dans la partie 'sortie écran' du programme
dk	Numéro de l'enregistrement à corriger
ds$	Chaîne devant recevoir un enregistrement
i	Index de comptage
k	Nombre d'enregistrements
ka	Numéro de l'enregistrement par lequel la 'recherche' commence
n$	Nom de l'auteur
r%	Code de retour pour les instructions BANK
s$	Texte recherché
sl	Longueur d'un champ de données
t$	Titre du livre
v$	Prénom de l'auteur
x$	Chaîne renvoyant la touche enfoncée

❐ Description du programme :

10	Commentaire.
20	Ouverture d'un fichier relatif avec une longueur d'enregistrement de 150 et initialisation de la variable entière r%.
30	Initialisation de la variable ds$ avec la longueur d'enregistrement indiquée.
40-130	Commentaire, passage en mode 40 colonnes et sortie du menu.
140-160	Entrée d'une valeur pour le choix du menu, passage en mode 80 colonnes et sauts dans le programme.

170-240	Entrée des enregistrements. A part les modifications suivantes, cette partie du programme est identique à la routine d'entrée du programme du chapitre 8.2. Mais l'entrée véritable se fait dans un sous-programme séparé (lignes 1330-1440) car cette partie du programme est devenue beaucoup plus développée du fait des longueurs prescrites pour les différents champs de données. Ce sous-programme peut aussi être utilisé par la partie 'corriger données' du programme. D'autre part les entrées sont écrites en ligne 230 directement dans le fichier RAM.
250-390	Sortie écran des enregistrements. Ici aussi, on procède de façon similaire au dernier chapitre. Cependant, alors que la version séquentielle comportait une boucle pour la sortie écran, on lit ici directement dans le fichier RAM (ligne 340). Par ailleurs, les sous-programmes des lignes 1260-1280 servent à diviser les enregistrements lus en fonction des longueurs prescrites pour les différents champs de données. Il faut noter également que les enregistrements entrés ou lus séquentiellement restent disponibles même si le programme est par exemple relancé avec RUN après une interruption pour une raison quelconque. Seule la valeur de la variable k (qui vaut automatiquement 0 après RUN) empêche alors la sortie de ces enregistrements dans cette partie du programme.
400-530	Sauvegarde (séquentielle) et chargement des enregistrements sur ou à partir de la disquette. Ici également on lit directement dans la RAM supplémentaire (ligne 430) ou on écrit directement dans la RAM supplémentaire (ligne 510).

540-960	Cette partie du programme recherche un texte entré par l'utilisateur. Suivant le choix du menu (lignes 540-600) on cherche chaque fois dans un des trois champs de données. L'entrée du texte recherché s'effectue dans le sous-programme des lignes 810-870. Le clavier est ici interrogé (ligne 820) en fonction de la longueur du champ d'enregistrement (variable sl). Les différentes entrées x$ sont sorties sur l'écran (ligne 840). Leur somme donne alors le texte recherché s$ (ligne 850). Le fait d'appuyer sur la touche COPY (numéro de touche 9) est compris comme l'entrée d'un caractère universel et un trait simple est alors sorti sur l'écran (ligne 830). Suivant le champ de données dans lequel s'effectue la recherche (voir la variable a) ce programme est appelé par la partie de programme des lignes 650-690 (champ 1, nom de l'auteur) ou par la partie des lignes 700-740 (champ 2, prénom) ou encore par la partie des lignes 750-790 (champ 3, titre du livre). La longueur de champ de données est déterminée dans ces parties de programme (variable sl), le texte recherché est chaque fois rempli de caractères universels pour les champs de données non pris en compte (voir les lignes 660, 710, 720 ou 770). Si s$ n'a pas été trouvé ou si une erreur est apparue lors de la sélection de banque (alors r% sera négatif), on saute à la ligne 800 où un commentaire approprié sera sorti et où sera effectué un retour au menu principal. Chaque fois que le texte recherché a été trouvé on appelle le sous-programme des lignes 880-960 pour sortir l'enregistrement correspondant. La variable r% indique alors le numéro d'enregistrement correspondant. Suivant le choix effectué par l'utilisateur du programme (voir la ligne 930) la recherche peut se poursuivre ou bien on saute au menu principal.

	Si la recherche doit se poursuivre, le paramètre ka, qui détermine le premier enregistrement concerné par l'opération de recherche, est augmenté de 1 dans les lignes 690, 740 ou 790, à moins que s$ n'ait été trouvé dans le dernier enregistrement (k-ième: IF r%<k). Suivant le cas, la recherche est donc poursuivie (retour) ou bien on saute à la ligne 800 dont nous avons déjà parlé. Indiquons encore que les variables ka et s$ sont définies en ligne 630 avec des valeurs de départ avant le début de l'opération de recherche.
970-1200	Correction ou suppression d'un enregistrement. On procède ici exactement comme dans les parties correspondantes de la gestion séquentielle du fichier de livres, compte tenu des modifications déjà indiquées. Indiquons simplement que lorsqu'on entre un enregistrement corrigé, on ne peut se contenter d'appuyer simplement sur la touche RETURN car cela serait compris dans le sous-programme concerné comme une interruption de l'entrée. Une telle interruption n'étant pas prévue pour la correction des données, cela provoquerait une diminution intempestive de la variable k.
1210-1240	Le programme se termine après une sortie de commentaire

1250-1280	Sous-programmes pour déterminer les champs de données. Comme on ne lit ou n'écrit dans le fichier RAM que des enregistrements entiers, ceux-ci doivent être divisés lorsqu'il s'agit de sortir séparément les différents champs de données. Le critère employé pour cette division est la longueur prescrite pour les différents champs de données. Il serait cependant également possible d'admettre une longueur variable des champs de données à condition que la longueur d'enregistrement reste fixe. Il faudrait alors définir un caractère de séparation permettant de distinguer les différents champs ("*" par exemple). L'enregistrement entier devrait toutefois comprendre alors deux caractères de plus (les caractères de séparation).
1290-1320	Sous-programmes pour les textes souvent utilisés.
1330-1440	Sous-programme pour remplir les champs de données. La routine d'entrée de la gestion séquentielle de fichier a été développée. Comme tous les champs de données ont une longueur fixe, les entrées trop courtes sont remplies avec des espaces (lignes 1360, 1390 ou 1420) et les entrées trop longues sont abrégées (lignes 1370, 1400 ou 1430).
1450-1490	Sous-programme 'attendre' et vidage de l'écran.

9. Programmes utilisateurs

9.1. Remarque préliminaire

Ce chapitre vous présentera encore quelques programmes pour vous montrer d'autres possibilités d'application. Ces programmes sont tous expliqués de manière détaillée. Ils sont organisés de façon à ce qu'on puisse facilement comprendre ce qui s'y passe. Cela s'est fait naturellement au détriment du style de programmation et de la vitesse de calcul. Nous vous invitons donc à 'bricoler' ces programmes, à les améliorer, à les compléter et à essayer de les rendres plus rapides et plus élégants...

Pour vous aider en cela, vous trouverez avec les explications de programme des conseils de programmation pour l'amélioration des programmes. Après avoir étudié les chapitres précédents, vous devriez donc être à même de tirer parti de ces remarques sur la technique de programmation. Les programmes suivants vous présentent par ailleurs une application pratique de certaines instructions BASIC importantes (WINDOW, PRINT USING, etc...). Ces exemples permettront, nous l'espérons, de dissiper les difficultés que pouvaient avoir certains d'entre vous à comprendre ces instructions.

Bien que l'excellent BASIC AMSTRAD (et peut-être également le présent ouvrage) donne vraiment envie de faire soi-même des programmes, nous donnons également des indications sur les performances de logiciels professionnels. Nous ne le faisons cependant que dans une mesure assez limitée puisque vous avez montré, en achetant ce livre, que vous ne tenez pas outre mesure à reprendre des idées toutes faites.

9.2. Analyse "cluster" des instructions BASIC

Avant d'en arriver aux programmes d'application proprement dits, nous allons vous expliquer plus en détail le programme

présenté au chapitre 3. Nous n'avons pas voulu vous présenter ni vous expliquer ce programme plus tôt car nous avons pensé qu'il aurait entraîné, au début de cet ouvrage, des problèmes de compréhension réels.

Le chapitre 3 vous donne une description détaillée du déroulement de l'opération. Nous essaierons plutôt ici de justifier ce que nous avons dit au chapitre 3.

Nous allons essayer d'illustrer la procédure suivie par un petit exemple. Les différentes instructions (c'est-à-dire les données de départ) seront ici représentées par des lettres.

- **Nous partirons des quatre enregistrements suivants :**

```
DATA a,3,c,q,p
DATA b,2,s,t
DATA c,4,r,s,t,u
DATA d,1,e
```

Il se passera alors la chose suivante. Lors du premier parcours (de la double boucle de programme des lignes 2080, 2090 à 2250, 2240, voyez le listing du programme) i=1 et j=2. Comme les instructions des deux premiers enregistrements sont différentes, j est augmenté de 1 et on compare le premier enregistrement au troisième. On trouve ici une 'ressemblance' puisque l'élément (l'instruction) 'c' figure dans les deux enregistrements. Le premier enregistrement est donc affecté au troisième qui contient donc maintenant les éléments (instructions) suivants :

```
c,r,s,t,u,a,q,p
```

Il faut noter que l'élément (l'instruction) 'c' n'apparaît bien sûr ici qu'une seule fois. La comparaison des premier et quatrième enregistrements, qui devrait maintenant être effectuée, n'a pas lieu puisque le premier enregistrement a déjà été affecté (z(1)=1). L'index i est donc augmenté de 1 et l'opération se poursuit avec comme résultat le fait que les deuxième et troisième enregistrements soient réunis. z(2) est alors égal à 2 et

le troisième enregistrement présente maintenant les éléments suivants :

```
c,r,s,t,u,a,q,p,b
```

Le troisième enregistrement est ensuite comparé au quatrième mais aucune 'ressemblance' ne peut être constatée. On a ainsi obtenu 2 catégories qui présentent les éléments suivants :

- 1ère catégorie : c,r,s,t,u,a,q,p,b
- 2de catégorie : d,e

L'un ou l'autre d'entre vous se demandera peut-être pourquoi l'enregistrement j n'est pas affecté à l'enregistrement i. Ce n'est pas possible car l'enregistrement ne participera plus au parcours qui va suivre et il ne sera donc pas possible de constater d'autres ressemblances entre l'enregistrement i nouvellement formé et les enregistrements restants. Rappelons que la comparaison effectuée plus haut entre les premier et second enregistrements n'a pas permis de constater de 'ressemblance'. Ce n'est que du fait de l'annexion du troisième enregistrement que les premier et second enregistrements de départ ont été réunis. On peut qualifier ce phénomène d'effet de 'chaîne' puisqu'une seule instruction suffit à affecter l'un à l'autre deux enregistrements déjà longs qui sont en réalité très différents. Si l'on veut exclure ces effets de chaîne, il faut choisir un autre critère de ressemblance. On pourrait par exemple décider qu'il faut deux instructions identiques pour effectuer un regroupement (une seule suffisait jusqu'ici). Sur le plan de la programmation, cela se présenterait ainsi :

```
2105 g=0
2135 g=g+1:IF g<2 THEN 2220
```

Avec cette version du programme, vous obtiendrez 60 catégories différentes. Ces catégories ne s'excluent pas mutuellement, c'est-à-dire que les mêmes instructions peuvent apparaître dans différentes catégories. Cela n'apporte rien à la clarté de l'exposé mais cela permet d'un autre côté d'identifier des instructions 'centrales', c'est-à-dire des instructions apparaissant souvent. On évite d'autre part, de cette manière,

que des instructions formant un groupe cohérent, telles que par exemple les instructions musicales, fassent partie d'une catégorie comprenant d'autres instructions. Mais comme nous souhaitons avoir le moins de catégories possible, nous ne vous présenterons pas les résultats fournis par cette seconde version.

Mais vous pouvez bien sûr très facilement essayer par vous-même ce que donne ce critère ou un autre critère de ressemblance.

Avant d'en venir au listing du programme, voici encore une proposition pour une autre application de ce programme. Supposez que vous vouliez organiser une colonie d'enfants et que vous ne sachiez pas comment répartir vos enfants par groupes. Vous pourriez alors effectuer cette répartition en vous aidant de votre CPC et du programme d'analyse 'cluster'. Il vous suffirait de demander à chaque enfant de vous indiquer le nom de ses amis et connaissances parmi ceux qui participent également à la colonie. Les résultats de ce questionnaire pourraient alors être entrés dans le programme comme données de départ. Vous pourriez vous assurer ainsi que les membres d'un groupe se connaissent. Le succès de la colonie serait alors 'programmé d'avance'.

```
10 REM Programme de classement de toutes les instructions du
BASIC AMSTRAD
20 REM n instructions differentes seront prises en compte par le
programme
30 n=168
40 DIM k$(n,65),ab(n),z(n)
50 CLS
60 LOCATE 14,12:PRINT"Je reflechis"
100 DATA ABS,1,SGN
110 DATA AFTER,3,EVERY,REMAIN,RETURN
120 DATA AND,3,OR,NOT,XOR
130 DATA ASC,1,CHR$
140 DATA ATN,5,SIN,COS,TAN,DEG,RAD
150 DATA AUTO,0
160 DATA BIN$,3,DEC$,HEX$,STR$
170 DATA BORDER,1,SPEED INK
180 DATA CALL,1,UNT
```

```
190 DATA CAT,0
200 DATA CHAIN,3,CHAIN MERGE,LOAD,MERGE
210 DATA CHAIN MERGE,4,CHAIN,DELETE,LOAD,MERGE
220 DATA CHR$,1,ASC
230 DATA CINT,5,CREAL,FIX,INT,ROUND,UNT
240 DATA CLEAR,0
250 DATA CLEAR INPUT,3,INKEY,INKEY$,JOY
260 DATA CLG,4,CLS,GRAPHICS PAPER,INK,ORIGIN
270 DATA CLOSEIN,2,EOF,OPENIN
280 DATA CLOSEOUT,1,OPENOUT
290 DATA CLS,4,CLG,INK,PAPER,WINDOW
300 DATA CONT,1,STOP
310 DATA COPYCHR$,1,LOCATE
320 DATA COS,4,ATN,DEG,RAD,SIN
330 DATA CREAL,1,CINT
340 DATA CURSOR,1,LOCATE
350 DATA DATA,2,READ,RESTORE
360 DATA DEC$,4,BIN$,HEX$,PRINT USING,STR$
370 DATA DEF FN,0
380 DATA DEFINT,2,DEFREAL,DEFSTR
390 DATA DEFREAL,2,DEFINT,DEFSTR
400 DATA DEFSTR,2,DEFINT,DEFREAL
410 DATA DEG,5,ATN,COS,RAD,SIN,TAN
420 DATA DELETE,2,CHAIN MERGE,RENUM
430 DATA DERR,5,ERL,ERR,ERROR,ON ERROR GOTO,RESUME
440 DATA DI,4,AFTER,EI,EVERY,REMAIN
450 DATA DIM,1,ERASE
460 DATA DRAW,3,DRAWR,GRAPHICS PEN,MASK
470 DATA DRAWR,3,DRAW,GRAPHICS PEN,MASK
480 DATA EDIT,2,AUTO,LIST
490 DATA EI,4,AFTER,DI,EVERY,REMAIN
500 DATA END,1,STOP
510 DATA ENT,2,ENV,SOUND
520 DATA ENV,2,ENT,SOUND
530 DATA EOF,2,OPENIN,CLOSEIN
540 DATA ERASE,1,DIM
550 DATA ERL,5,DERR,ERR,ERROR,ON ERROR GOTO,RESUME
560 DATA ERR,5,DERR,ERL,ERROR,ON ERROR GOTO,RESUME
570 DATA ERROR,4,ERL,ERR,ON ERROR GOTO,RESUME
580 DATA EVERY,2,AFTER,REMAIN
590 DATA EXP,1,LOG
```

```
600 DATA FILL,1,GRAPHICS PEN
610 DATA FIX,3,CINT,INT,ROUND
620 DATA FOR,3,NEXT,STEP,TO
630 DATA FRAME,2,TAG,TAGOFF
640 DATA FRE,2,HIMEM,MEMORY
650 DATA GOSUB,1,RETURN
660 DATA GOTO,0
670 DATA GRAPHICS PAPER,6,CLG,GRAPHICS PEN,INK,MASK,TAG,TAGOFF
680 DATA GRAPHICS PEN,5,GRAPHICS PAPER,INK,MASK,TAG,TAGOFF
690 DATA HEX$,4,BIN$,DEC$,STR$,UNT
700 DATA HIMEM,4,FRE,MEMORY,SYMBOL,SYMBOL AFTER
710 DATA IF,3,ELSE,GOTO,THEN
720 DATA INK,5,GRAPHICS PAPER,GRAPHICS PEN,PAPER,PEN,SPEED INK
730 DATA INKEY,3,CLEAR INPUT,INKEY$,JOY
740 DATA INKEY$,1,CLEAR INPUT
750 DATA INP,2,OUT,WAIT
760 DATA INPUT,1,LINE INPUT
770 DATA INSTR,0
780 DATA INT,3,CINT,FIX,ROUND
790 DATA JOY,2,CLEAR INPUT,INKEY
800 DATA KEY,1,KEY DEF
810 DATA KEY DEF,2,KEY,SPEED KEY
820 DATA LEFT$,2,MID$,RIGHT$
830 DATA LEN,0
840 DATA LET,0
850 DATA LINE INPUT,1,INPUT
860 DATA LIST,0
870 DATA LOAD,5,CHAIN,CHAIN MERGE,MERGE,RUN,SAVE
880 DATA LOCATE,1,WINDOW
890 DATA LOG,2,EXP,LOG 10
900 DATA LOG 10,2,EXP,LOG
910 DATA LOWER$,1,UPPER$
920 DATA MASK,4,DRAW,DRAWR,GRAPHICS PAPER,GRAPHICS PEN
930 DATA MAX,1,MIN
940 DATA MEMORY,4,FRE,HIMEM,SYMBOL,SYMBOL AFTER
950 DATA MERGE,3,CHAIN,CHAIN MERGE,LOAD
960 DATA MID$,2,LEFT$,RIGHT$
970 DATA MIN,1,MAX
980 DATA MOD,0
990 DATA MODE,2,ORIGIN,WINDOW
1000 DATA MOVE,4,MOVER,ORIGIN,XPOS,YPOS
```

```
1010 DATA MOVER,4,MOVE,ORIGIN,XPOS,YPOS
1020 DATA NEW,0
1030 DATA NEXT,3,FOR,STEP,TO
1040 DATA NOT,3,AND,OR,XOR
1050 DATA ON BREAK CONT,2,ON BREAK GOSUB,ON BREAK STOP
1060 DATA ON BREAK GOSUB,3,ON BREAK CONT,ON BREAK STOP,RETURN
1070 DATA ON BREAK STOP,2,ON BREAK CONT,ON BREAK GOSUB
1080 DATA ON ERROR GOTO,5,DERR,ERL,ERR,ERROR,RESUME
1090 DATA ON GOSUB,1,RETURN
1100 DATA ON GOTO,0
1110 DATA ON SQ GOSUB,3,RETURN,SOUND,SQ
1120 DATA OPENIN,2,CLOSEIN,EOF
1130 DATA OPENOUT,1,CLOSEOUT
1140 DATA OR,3,AND,NOT,XOR
1150 DATA ORIGIN,1,WINDOW
1160 DATA OUT,2,INP,WAIT
1170 DATA PAPER,3,GRAPHICS PAPER,INK,PEN
1180 DATA PEEK,1,POKE
1190 DATA PEN,1,PAPER
1200 DATA PI,2,DEG,RAD
1210 DATA PLOT,2,GRAPHICS PEN,PLOTR
1220 DATA PLOTR,2,GRAPHICS PEN,PLOT
1230 DATA POKE,1,PEEK
1240 DATA POS,2,VPOS,WINDOW
1250 DATA PRINT,4,SPC,TAB,USING,ZONE
1260 DATA RAD,5,ATN,COS,DEG,SIN,TAN
1270 DATA RANDOMIZE,1,RND
1280 DATA READ,2,DATA,RESTORE
1290 DATA RELEASE,1,SOUND
1300 DATA REM,0
1310 DATA REMAIN,4,AFTER,DI,EI,EVERY
1320 DATA RENUM,2,DELETE,LIST
1330 DATA RESTORE,2,DATA,READ
1340 DATA RESUME, 6, DERR,ERL,ERR,ERROR,ON ERROR GOTO,
RESUME NEXT
1350 DATA RESUME NEXT,6, DERR,ERL,ERR,ERROR,ON ERROR GOTO,
RESUME
1360 DATA RETURN,1,GOSUB
1370 DATA RIGHT$,2,MID$,LEFT$
1380 DATA RND,1,RANDOMIZE
1390 DATA ROUND,4,ABS,CINT,FIX,INT
```

```
1400 DATA RUN,4,LOAD,CONT,END,STOP
1410 DATA SAVE,5,CHAIN,CHAIN MERGE,LOAD,MERGE,RUN
1420 DATA SGN,1,ABS
1430 DATA SIN,5,ATN,COS,DEG,RAD,TAN
1440 DATA SOUND,5,ENT,ENV,ON SQ GOSUB,RELEASE,SQ
1450 DATA SPACE$,3,SPC,STRING$,TAB
1460 DATA SPEED INK,2,BORDER,INK
1470 DATA SPEED KEY,1,KEY DEF
1480 DATA SPEED WRITE,2,OPENOUT,SAVE
1490 DATA SQ,2,ON SQ GOSUB,SOUND
1500 DATA SQR,0
1510 DATA STOP,2,CONT,END
1520 DATA STR$,4,BIN$,DEC$,HEX$,VAL
1530 DATA STRING$,1,SPACE$
1540 DATA SYMBOL,1,SYMBOL AFTER
1550 DATA SYMBOL AFTER,3,HIMEM,MEMORY,SYMBOL
1560 DATA TAG,1,TAGOFF
1570 DATA TAGOFF,1,TAG
1580 DATA TAN,5,ATN,COS,DEG,RAD,SIN
1590 DATA TEST,5,MOVE,MOVER,TESTR,XPOS,YPOS
1600 DATA TESTR,5,MOVE,MOVER,TEST,XPOS,YPOS
1610 DATA TIME,4,AFTER,EVERY,WEND,WHILE
1620 DATA TRON,1,TROFF
1630 DATA UNT,4,CINT,FIX,INT,ROUND
1640 DATA UPPER$,1,LOWER$
1650 DATA VAL,1,STR$
1660 DATA VPOS,2,POS,WINDOW
1670 DATA WAIT,2,INP,OUT
1680 DATA WEND,2,TIME,WHILE
1690 DATA WHILE,2,TIME,WEND
1700 DATA WIDTH,1,POS
1710 DATA WINDOW,1,WINDOW SWAP
1720 DATA WINDOW SWAP,1,WINDOW
1730 DATA WRITE,2,INPUT,LINE INPUT
1740 DATA XOR,3,AND,OR,NOT
1750 DATA XPOS,4,MOVE,MOVER,ORIGIN,YPOS
1760 DATA YPOS,4,MOVE,MOVER,ORIGIN,XPOS
1770 DATA ZONE,1,PRINT
2000 REM Lecture des donnees
2010 FOR i=1 TO n
2020 READ k$(i,0),ab(i)
```

```
2030 FOR j=1 TO ab(i)
2040 READ k$(i,j)
2050 NEXT j
2060 NEXT i
2070 REM Le classement effectif
2080 FOR i=1 TO n-1
2090 FOR j=i+1 TO n
2100 IF z(i)=1 THEN 2250
2110 FOR i1=0 TO ab(i)
2120 FOR j1=0 TO ab(j)
2130 IF k$(i,i1)<>k$(j,j1) THEN 2220
2140 ij=0:ab=ab(j)
2150 z(i)=1
2160 FOR l=0 TO ab
2170 IF k$(i,ij)<>k$(j,l) THEN 2190
2180 IF ij<ab(i) THEN ij=ij+1:GOTO 2160 ELSE 2240
2190 NEXT l
2200 ab(j)=ab(j)+1:k$(j,ab(j))=k$(i,ij)
2210 IF ij<ab(i) THEN ij=ij+1:GOTO 2160 ELSE 2240
2220 NEXT j1
2230 NEXT i1
2240 NEXT j
2250 NEXT i
2260 REM Sortie des resultats
2270 FOR i=1 TO n
2280 IF z(i)=1 THEN 2360
2290 k=k+1
2300 CLS:PRINT "Categorie"k:PRINT
2310 FOR j=0 TO ab(i)
2320 PRINT k$(i,j),
2330 NEXT j
2340 LOCATE 7,25:PRINT"Frappez une touche S.V.P."
2350 a$=INKEY$:IF a$="" THEN 2350 ELSE CLS
2360 NEXT i
2370 PRINT"Fin du programme":END
```

❐ Liste de variables :

ab(i)	Nombre d'instructions apparentées (pour chaque enregistrement).
ab	Nombre des instructions apparentées de l'enregistrement auxquelles est affecté un autre enregistrement.
i	Index de comptage.
i1	Index de comptage.
ij	Second index pour k$ dans l'opération d'affectation.
j	Index de comptage.
j1	Index de comptage.
k	Numéro actuel d'une catégorie lors de la sortie de résultats.
k$(i,j)	Les instructions (index j) d'un enregistrement i.
l	Index de comptage.
n	Nombre d'enregistrements.
z(i)	Variable désignant un enregistrement (z(i) égale 1 si l'enregistrement i est affecté, sinon 0).

❐ Description du programme :

10-20	Commentaire.
30-40	Définition de la variable "n" et réservation de place mémoire.
50-60	Vidage de l'écran et sortie de commentaire pendant l'opération de calcul.
100-1770	Préparation des données. Structure des instructions DATA : Instruction DATA, nombre d'instructions apparentées, instructions apparentées.
2000-2060	Lecture des données. L'instruction concernée est chaque fois placée dans le tableau k$(i,0) alors que les tableaux k$(i,j), avec j=1-nombre d'instructions apparentées, contiennent les instructions apparentées.

2070-2260	Réalisation des affectations. Chaque enregistrement est comparé à chaque autre enregistrement dans la boucle de programme (lignes 2080, 2090 à 2250, 2240) si l'enregistrement i n'est pas encore affecté (voir la ligne 2100). Une comparaison consiste à examiner toutes les instructions de chacun des deux enregistrements confrontés, c'est-à-dire que chaque instruction d'un enregistrement est comparée à chaque instruction de l'autre enregistrement (double boucle des lignes 2110, 2120 à 2230, 2220). L'examen véritable s'effectue en ligne 2130 et les lignes 2140 à 2210 ne sont exécutées que si une instruction figure aussi bien dans l'un que dans l'autre enregistrement. Les instructions de l'enregistrement i sont en effet affectées dans ce cas aux instructions de l'enregistrement j (ligne 2200) et l'enregistrement i est marqué en conséquence (ligne 2150). Une instruction n'est toutefois affectée que si elle ne figure pas encore dans l'enregistrement j (examen fait dans les lignes 2160 à 2190, la variable 'ab' est à cet effet fixée en ligne 2140 égale à 'ab(j)' car 'ab(j)' se modifie au cours de la procédure d'affectation (voyez la ligne 2200)). La ligne 2200 (et donc aussi la ligne 2210) n'est par conséquent atteinte, pour chacune des instructions de l'enregistrement i, que si la comparaison de la ligne 2170 s'est conclue négativement. Dans les lignes 2180 et 2210, l'index 'ij' de l'enregistrement i (qui est fixé sur 0 en ligne 2140 pour chaque parcours) est augmenté de 1, si l'enregistrement i contient encore d'autres instructions (IF ij < ab(i)), et on saute à l'examen de cette autre instruction. S'il n'y a plus d'autres instructions dans l'enregistrement i, on passe au prochain enregistrement 'j' (saut à la ligne 2240).

2260-2370	Sortie de résultats : la boucle de programme (lignes 2270 à 2360) sort les instructions des enregistrements restants, c'est-à-dire des enregistrements qui n'ont pas été regroupés. Comme ces enregistrements se composent en général de plusieurs des enregistrements du départ, ils sont qualifiés de 'catégories'. Après chaque sortie des instructions d'une catégorie, les lignes 2340 et 2350 attendent qu'on frappe sur une touche pour éviter un scrolling (glissement) trop rapide de l'écran. Une fois que tous les résultats ont été sortis, le programme se termine en ligne 2370.

9.3. Remboursement de dettes

Ce chapitre aborde le problème des dettes. Si vous demandez par exemple un crédit de 100 000 F, votre banque vous compte en général un agio. Cet agio doit être décompté mais il n'est pas payé. Il est simplement retenu par votre banque comme une sorte de taxe. Lors du calcul des intérêts et du remboursement de la dette vous devez donc partir de la dette nominale et non des sommes versées.

Lors de l'acceptation du crédit, vous convenez avec votre banque d'un taux d'intérêt et d'un remboursement initial (également en pourcentage du montant nominal du crédit). C'est de ces deux éléments que se composent les remboursements que vous aurez à effectuer chaque mois ou chaque trimestre. Avec un taux d'intérêt de 6% et un remboursement initial de 2%, cela représente 2000 F par trimestre ou 8000 F par an. Ces remboursements ne varient pas pendant toute la durée du crédit. La partie extinction de la dette de ces remboursements augmente toutefois constamment en même temps que diminue la partie correspondant aux intérêts puisque le capital produisant des intérêts se réduit d'année en année. C'est pourquoi le remboursement du prêt ne s'effectue

pas dans l'exemple ci-dessus en 50 ans mais seulement en environ 23 ans (pour un taux d'intérêt inchangé).

Les offres de crédit diffèrent notamment dans la durée du taux d'intérêt fixé conventionnellement. Une fois cette durée écoulée, le taux d'intérêt peut être augmenté ou diminué en fonction de l'état du marché. Mais vous pouvez alors dans ce cas rembourser immédiatement le solde de la dette si vous le souhaitez, que ce soit avec vos propres moyens ou que ce soit avec de l'argent à nouveau emprunté.

Il peut donc être très important pour vous de connaître le montant du reliquat de la dette. Ce montant peut être calculé à l'aide d'un plan de remboursement qui reconstitue la gestion de compte d'un institut de crédit. Il faut bien sûr prendre en compte tous les paramètres du crédit tels que remboursements, capital produisant des intérêts, intérêt, extinction de la dette et reliquat de la dette. C'est un tel plan de remboursement que produit le programme suivant. On vous montre en même temps comment on peut avec le CPC créer des tableaux, avec le mode 80 caractères et l'instruction PRINT USING.

Quelques remarques sont cependant nécessaires au sujet de ce plan de remboursement :

1) On part comme base de calcul de la dette nominale et non du montant effectivement payé.

2) C'est cette dette nominale qui produit des intérêts et qui est remboursée.

3) Bien que les remboursements soient payés régulièrement (tous les trimestres dans notre programme), l'intérêt est calculé pour chaque année pleine par rapport à un montant de capital inchangé. Ce n'est qu'après écoulement d'une année pleine que le capital produisant des intérêts diminue.

4) Le reliquat de la dette se réduit par contre avec chaque remboursement à raison du montant correspondant à l'extinction de la dette. Au bout d'une année pleine, ce reliquat est égal au capital produisant intérêt.

5) Les mouvements de compte sont calculés en fin d'année. Les intérêts, les paiements d'extinction de la dette et de remboursement sont additionnés et le capital est diminué du montant correspondant à l'extinction de la dette. Le reliquat de la dette n'est pas modifié par ces calculs puisqu'il diminue avec chaque remboursement (bien sûr uniquement lorsque cela correspond à l'extinction de la dette).

6) Le plan de remboursement se poursuit jusqu'à ce que le reliquat de la dette soit inférieur au remboursement, ce qui est signalé par un message dans notre programme.

7) Pour optimiser la sortie à l'écran nous avons renoncé dans ce programme à sortir isolément le message "Frappez une touche S.V.P." lors de la création du tableau. Il faut par ailleurs tenir compte du nombre de chiffres dans l'instruction PRINT USING. Si vous entrez des montants de l'ordre du milliard, la sortie du tableau sera perturbée.

```
10 REM Plan de remboursement
20 CLS
30 PRINT TAB(7)"Programme pour etablir"
40 PRINT TAB(7)"un plan de remboursement"
50 PRINT:PRINT:PRINT:PRINT:PRINT
60 PRINT TAB(11) "Bernd Kowal, 1985"
70 GOSUB 640
80 INPUT"Montant (nominal) du credit ";Kb:PRINT
90 INPUT"Taux d'interet nominal   ";Zs:PRINT
100 INPUT"Remboursement initial en % ";t:PRINT
110 PRINT"Entree de la date de versement :"
120 INPUT"Jour  ";tg
130 INPUT"Mois  ";m
140 INPUT"Annee ";j$:PRINT
150 j$=RIGHT$(j$,2):j=VAL(j$)
160 PRINT"C'est donc le";tg;".";m;".";j
170 PRINT"qui sera pris comme date de versement."
180 GOSUB 640:CLS:MODE 2
190 GOSUB 550
200 k1=1:k2=5:R=0:Zi=0:Ti=0:Rs=Kb
210 FOR i=k1 TO k2
220 IF i=k1 THEN 260
```

```
230 IF m>9 AND j=99 THEN j=0:GOTO 250
240 IF m>9 THEN j=j+1
250 IF m>9 THEN m=m-9 ELSE m=m+3
260 PRINT;USING"##.";tg;:PRINT;USING"##.";m;
270 PRINT;USING"##";j;
280 IF i=1 THEN PRINT TAB(11)"Versement";:GOTO 310
290 IF i=k1 THEN PRINT TAB(11)"Liquidation";:GOTO 310
300 PRINT TAB(11)"Traite ";i-1;
310 PRINT TAB(23);USING"#####.##";R;
320 PRINT;USING"   #######.##";Kb;
330 PRINT;USING"   #####.##";Zi;
340 PRINT;USING"   #####.##";Ti;
350 PRINT;USING"   #######.##";Rs
360 IF i>k1 THEN 410
370 IF i=1 THEN R=Kb*(Zs+t)/400 ELSE R=R/4
380 PRINT
390 Zi=Kb*Zs/400
400 Ti=R-Zi
410 Rs=Rs-Ti
420 IF Rs<0 THEN 510
430 IF i<k2 THEN 460
440 R=R*4:Zi=Zi*4:Ti=Ti*4
450 Kb=Kb-Ti:Rs=Rs+Ti/4
460 IF zae=3 AND k1=i THEN zae=0:GOSUB 640:GOSUB 560
470 NEXT i
480 zae=zae+1
490 k1=k2:k2=k2+4
500 GOTO 210
510 PRINT:PRINT"Avec le remboursement de";
520 PRINT INT((Rs+Ti)*100+0.5)/100;"FF, ";
530 PRINT"la dette est eteinte."
540 GOSUB 640:MODE 1:END
550 REM SP Titre du tableau
560 PRINT"Date       Operation    Traite    Capital";
570 PRINT"       Interet    Rembours.    Reste"
580 PRINT TAB(25)"en FF";TAB(35)"produisant";
590 PRINT TAB(48)"en FF";TAB(58)"en FF";TAB(71)"du en FF"
600 PRINT TAB(35)"interet";TAB(71)"en FF"
610 PRINT TAB(35)"en FF"
620 PLOT 0,330:DRAW 620,330:PRINT
```

```
630 RETURN
640 REM SP Attendre
650 LOCATE 7,25
660 IF i=0 THEN PRINT"Frappez une touche S.V.P. !"
670 x$=INKEY$
680 IF x$="" THEN 670
690 CLS:RETURN
```

❐ Liste de variables :

i	index de comptage.
j	année.
j$	tableau d'entrée pour j.
k1	valeur initiale de la boucle.
k2	valeur finale de la boucle.
Kb	montant nominal du crédit, soit le capital produisant des intérêts.
m	mois.
R	remboursement en francs.
Rs	reliquat de la dette en francs.
t	remboursement initial en %.
tg	jour.
Ti	extinction de la dette en francs.
zae	variable de comptage.
Zi	intérêt en francs.
Zs	taux d'intérêt nominal en %.

❐ Description du programme :

Lignes 10-70 :	Titre.
Lignes 80-170 :	Entrée des valeurs initiales. L'indication de l'année est traitée comme une variable alphanumérique (=chaîne de caractères) de façon à pouvoir en ligne 150 ne convertir que les deux derniers caractères en une expression numérique. Cette méthode un peu compliquée a pour effet de n'utiliser que "86" comme année lors de la création du tableau si l'entrée était par exemple "1986".
Ligne 180 :	On saute au sous-programme "attendre", l'écran est vidé et on change le mode écran (80 colonnes).
Ligne 190 :	Saut au sous-programme (ligne 550) pour sortir le titre du tableau.
Ligne 200 :	Affectation des valeurs initiales aux variables pour le premier parcours de la boucle.
Lignes 210-470 :	Constitution d'une boucle de calcul des différentes grandeurs et de sortie des résultats sous forme de tableau pour une période (=1 an). Lors de chaque premier parcours on prend en compte l'opération "Paiement" ou "Solde".
220-250 :	Pour les parcours 2 à 5 de chaque période (c'est alors que les remboursements sont payés), modification de la date.
260-270 :	Sortie de la date.
280-300 :	Sortie de l'opération dont il s'agit (1er parcours de la première période : paiement, 1er parcours des périodes suivantes : solde, parcours 2 à 5 de toutes les périodes : remboursement).
310-350 :	Sortie des autres valeurs du tableau.

360-400 :	Lors de chaque premier parcours, calcul des remboursement, intérêts et extinction de la dette (montants constants pour chaque période) et sortie d'une ligne vide. A noter en ligne 370 que le remboursement n'est en fait calculé que lors du premier parcours de la première période. Pour le reste, le dernier solde est simplement divisé par 4.
430-450 :	Lors de chaque dernier parcours, sont calculés les remboursement, intérêt, capital produisant inté-rêts et reliquat de la dette pour le solde.
460 :	Sortie des résultats pour 3 périodes (zae=3), y compris solde (kl=i, premier parcours), remise à 0 de l'index de comptage i et saut aux sous-programme "attendre" et "titre du tableau".
Lignes 480-500 :	A la fin du parcours de la boucle pour une période, l'index de comptage est augmenté de 1, les valeurs initiale et finale de la boucle reçoivent de nouvelles valeurs et on saute à la ligne 210 (nouveau parcours de la boucle).
Lignes 510-530 :	Sortie d'un commentaire concernant le reliquat de la dette lorsqu'il ne reste plus de remboursement à effectuer.
Ligne 540 :	Saut au sous-programme "attendre", passage en mode 40 colonnes et fin du programme.
Lignes 550-630 :	Sous-programme "titre du tableau". Le titre est souligné en ligne 620. Le curseur graphique est amené avec l'instruction PLOT dans l'emplacement voulu et la ligne est dessinée avec l'instruction DRAW.
Lignes 640-690 :	Sous-programme "attendre". Le commentaire "Frappez une touche S.V.P." n'est sorti que si la création du tableau n'a pas encore commencé.

❑ **Résultats du programme :**

Lancez le programme et entrez un montant de crédit nominal de 10000 F, un intérêt annuel de 7% et un remboursement initial de 3% avec versement de l'argent le 30 12 1986. Vous verrez alors que pour un versement trimestriel de 225 F, vous devrez verser 88 remboursements jusqu'au 30 12 08 (donc exactement 22 ans) et qu'il faut en outre payer un reliquat de 198.85 F. Vous rembourserez donc au total 19998.85.

9.4. Budget rationnel

Disons-le tout de suite : nous ne présentons pas dans ce chapitre de programme de gestion de comptabilité familiale. Un budget domestique ne se gère de toute façon pas comme une PME.

Nous allons plutôt vous donner une aide technique en programmation pour vous permettre de conserver ou d'essayer de conserver le contrôle de tous les coûts qui grèvent votre gestion domestique.

Plus vous employez votre argent de façon rationnelle et plus vous pouvez vous permettre de choses. Cette opinion très répandue est certainement fondée. Mais que veut dire rationnel ? Pour le consommateur, et nous sommes tous des consommateurs, il est souvent à peine possible de juger de la qualité d'un produit. D'autre part les prix bougent sans cesse dans notre système économique. Il n'est donc pas si simple d'avoir un comportement rationnel.

Il est donc peut-être judicieux de se demander pour quoi est dépensé l'argent du ménage. Si vous ne vous êtes jamais encore posé cette question, vous risquez d'être fort étonné une fois que vous découvrirez comment vos dépenses se répartissent entre les différentes catégories que sont les vêtements, l'alimentation, les voyages, etc. Un tel contrôle de vos sorties d'argent comprend bien sûr une grande masse d'informations. C'est pourquoi l'emploi d'un CPC peut être fort utile pour la gestion de ces données.

Le programme suivant se charge de cette tâche en offrant les fonctions suivantes :

▾ Entrée de sorties d'argent (avec les variables "nom marchandise", "montant francs", "catégorie" et "date").

▾ Constitution de catégories (jusqu'à 10 catégories peuvent être constituées).

▾ Pour les sorties d'argent et les catégories :

  - lecture.

  - sauvegarde.

  - examen (sous forme de tableau).

▾ calcul (les montants en francs par catégorie sont sortis pour n'importe quelle période. La représentation de la répartition en pourcentage des différentes catégories se fait sous la forme d'un histogramme.).

▾ Fin du travail.

Le programme ne gère donc pas seulement les différentes informations mais il prend également en charge la tâche importante qui consiste à effectuer des calculs. Vous pouvez ainsi constater en fin de mois ou en fin d'année combien vous avez dépensé d'argent pour chaque catégorie de dépenses.

Comme vous ne pouvez constituer que 10 catégories (un plus grand nombre de catégories nuirait à la clarté du tableau), il est recommandé de créer une catégorie "divers" pour éviter des problèmes insolubles de classement. Vous devez effectuer une lecture AVANT d'entrer de nouvelles données car sinon les informations que vous avez entrées avant le chargement seront effacées.

Il y a place en tout dans le programme pour un maximum de 1000 entrées. Vous pouvez bien sûr réhausser cette limite (lignes 80-90 mais attention à la capacité mémoire de votre ordinateur) mais il serait certainement plus judicieux d'établir un bilan final pour chaque mois ou chaque année.

Pour le reste, le programme est largement auto-explicatif. Nous avons renoncé après la sortie de l'histogramme au message "Frappez une touche S.V.P." car la place disponible sur l'écran est intégralement employée pour le graphique et il s'agissait d'éviter un glissement de l'écran vers le haut.

```
10 REM Budget domestique
20 CLS
30 PRINT"     Programme de controle"
40 PRINT"     de vos depenses"
50 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
60 PRINT TAB(9) "Bernd Kowal, 1985"
70 GOSUB 2060
80 DIM wb$(1000),dm(1000),k(1000)
90 DIM tg(1000),mo(1000),ja(1000),ja$(1000)
100 REM
110 REM Menu
120 REM
130 PRINT TAB(17)"Menu":PRINT:PRINT
140 PRINT TAB(33)"Entree":PRINT
150 PRINT"Entrer depenses";TAB(36)"1":PRINT
160 PRINT"Creer categories";TAB(36)"2":PRINT
170 PRINT"Depenses et categories :":PRINT
180 PRINT" charger";TAB(36)"3":PRINT
190 PRINT" sauvegarder";TAB(36)"4":PRINT
200 PRINT" examiner";TAB(36)"5":PRINT
210 PRINT" calculer";TAB(36)"6"
220 PRINT:PRINT"Fin du travail";TAB(36)"7"
230 PRINT:PRINT:INPUT"Votre choix ";a:CLS
240 IF a<1 OR a>7 THEN CLS:GOTO 130
250 ON a GOTO 270,620,780,920,1080,1240,1930
260 REM
270 REM Entrer les depenses
280 REM
290 IF ka>0 THEN 320
300 PRINT"Vous devez d'abord creer des categories."
310 GOSUB 2050:GOTO 620
320 WINDOW #2,20,40,1,25
330 PRINT #2,"Categories":PRINT #2
340 FOR i=1 TO ka
350 PRINT #2,i;".";k$(i)
```

```
360 NEXT i
370 WINDOW #1,1,19,1,25
380 ga=ga+1
390 PRINT #1,"L'entree s'acheve"
400 PRINT #1,"si vous entrez"
410 PRINT #1,"un 0 comme nom"
420 PRINT #1,"de marchandise.":PRINT #1
430 PRINT #1,"Veuillez entrer"
440 PRINT #1,"les informations"
450 PRINT #1,"suivantes:":PRINT #1:PRINT #1
460 PRINT #1,"Entree";ga;":":PRINT #1
470 PRINT #1,"Nom de marchandise"
480 INPUT #1,wb$(ga):PRINT #1
490 IF wb$(ga)="0" THEN ga=ga-1:CLS:GOTO 110
500 PRINT #1,"Montant en FF"
510 INPUT #1,dm(ga):PRINT #1
520 PRINT #1,"Categorie"
530 INPUT #1,"Nr. ";k(ga):PRINT #1
540 IF k(ga)<1 OR k(ga)>ka THEN 520
550 PRINT #1,"Date"
560 INPUT #1,"Jour   ";tg(ga)
570 INPUT #1,"Mois   ";mo(ga)
580 INPUT #1,"Annee  ";ja$(ga)
590 ja$(ga)=RIGHT$(ja$(ga),2):ja(ga)=VAL(ja$(ga))
600 CLS #1:GOTO 380
610 REM
620 REM Creer des categories
630 REM
640 IF ka=0 THEN 690
650 PRINT"Les categories suivantes existent deja:"
660 PRINT:FOR i=1 TO ka
670 PRINT "categorie";i;":";k$(i)
680 NEXT i:PRINT:PRINT
690 PRINT"Vous pouvez creer 10 categories maximum."
700 PRINT"Entrez un 0 si vous ne voulez plus"
710 PRINT"creer de nouvelle categorie.":PRINT
720 ka=ka+1
730 IF ka=11 THEN ka=ka-1:GOSUB 2050:GOTO 110
740 PRINT "Categorie";ka;:INPUT k$(ka)
750 IF k$(ka)="0" THEN ka=ka-1:CLS:GOTO 110
760 GOTO 720
```

```
770 REM
780 REM Charger depenses et categories
790 REM
800 INPUT"Nom du fichier ";n$:OPENIN n$
810 INPUT #9,ga,ka
820 FOR i=1 TO ga
830 INPUT #9,wb$(i)
840 INPUT #9,dm(i),k(i),tg(i),mo(i),ja(i)
850 NEXT i
860 FOR i=1 TO ka
870 INPUT #9,k$(i)
880 NEXT i
890 CLOSEIN
900 CLS:GOTO 110
910 REM
920 REM Sauvegarder depenses et categories
930 REM
940 PRINT"Sous quel nom les informations doivent-"
950 INPUT"elles etre sauvegardees";n$
960 OPENOUT n$
970 PRINT #9,ga,ka
980 FOR i=1 TO ga
990 PRINT #9,wb$(i)
1000 PRINT #9,dm(i),k(i),tg(i),mo(i),ja(i)
1010 NEXT i
1020 FOR i=1 TO ka
1030 PRINT #9,k$(i)
1040 NEXT i
1050 CLOSEOUT
1060 CLS:GOTO 110
1070 REM
1080 REM Examiner depenses et categories
1090 REM
1100 MODE 2:af=1:h=20
1110 PRINT"No d'entree   Nom de marchandise  ";
1120 PRINT"Montant FF    Categorie         Date"
1130 PRINT:FOR i=af TO ga
1140 PRINT USING "   #####    ";i;
1150 PRINT USING "   \                 \";wb$(i);
1160 PRINT USING "    ######.
##";dm(i);
```

```
1170 PRINT USING "     \           \      ";k$(k(i));
1180 PRINT USING "##.";tg(i);mo(i);
1190 PRINT USING "##";ja(i)
1200 IF i=h THEN h=h+20:af=af+20:GOSUB 2060:GOTO 1110
1210 NEXT i
1220 GOSUB 2050:MODE 1:GOTO 110
1230 REM
1240 REM Calculs
1250 REM
1260 PRINT"Veuillez entrer la periode que doivent"
1270 PRINT"concerner les calculs.":PRINT
1280 PRINT"de :";TAB(18)"a :":PRINT
1290 PRINT:PRINT"Jour";TAB(18)"Jour":PRINT
1300 PRINT"Mois";TAB(18)"Mois":PRINT
1310 PRINT"Annee";TAB(18)"Annee"
1320 LOCATE 7,7:INPUT at
1330 LOCATE 7,9:INPUT am
1340 LOCATE 7,11:INPUT aja$
1350 LOCATE 24,7:INPUT et
1360 LOCATE 24,9:INPUT em
1370 LOCATE 24,11:INPUT eja$
1380 aja$=RIGHT$(aja$,2):aja=VAL(aja$)
1390 eja$=RIGHT$(eja$,2):eja=VAL(eja$)
1400 PRINT:PRINT:PRINT TAB(8)"Je calcule !"
1410 su=0:FOR i=1 TO ka
1420 ks(i)=0:NEXT i
1430 FOR i=1 TO ga
1440 IF ja(i)<aja OR ja(i)>eja THEN 1510
1450 IF ja(i)>aja AND ja(i)<eja THEN 1490
1460 IF mo(i)<am OR mo(i)>em THEN 1510
1470 IF mo(i)>am AND mo(i)<em THEN 1490
1480 IF tg(i)<at OR tg(i)>et THEN 1510
1490 ks(k(i))=ks(k(i))+dm(i)
1500 su=su+dm(i)
1510 NEXT i
1520 CLS:PRINT"Dans cette periode, vous avez depense"
1530 PRINT"au total";INT(su*100+0.5)/100;"FF."
1540 PRINT:PRINT
1550 PRINT"Pour les differentes categories, cela"
1560 PRINT"donne l'image suivante :":PRINT
1570 PRINT"No de la   Nom de la";TAB(35)"FF"
```

```
1580 PRINT"Categorie  Categorie":PRINT
1590 FOR i=1 TO ka
1600 PRINT USING "#####";i;
1610 PRINT USING "     \               \";k$(i);
1620 z=INT(ks(i)*100+0.5)/100
1630 PRINT TAB(30) USING "#######.##";z
1640 NEXT i
1650 GOSUB 2050
1660 PRINT"Representation graphique de la"
1670 PRINT"repartition en pourcentages arrondis"
1680 PRINT"des differentes categories de depense:"
1690 pm=0
1700 FOR i=1 TO ka
1710 p(i)=ks(i)*100/su
1720 IF p(i)>pm THEN pm=p(i)
1730 NEXT i
1740 FOR i=1 TO ka
1750 pp(i)=p(i)*100/pm
1760 NEXT i
1770 PLOT 17,17:DRAW 17,320
1780 PLOT 17,17:DRAW 520,17
1790 LOCATE 1,5:PRINT"%"
1800 LOCATE 34,25:PRINT"Categ."
1810 j=0
1820 FOR i=3 TO ka*3 STEP 3
1830 j=j+1
1840 LOCATE i,25:PRINT j
1850 x=ROUND(18*pp(j)/100)
1860 FOR l=24 TO 24-x STEP -1
1870 LOCATE i+1,l:PRINT CHR$(143)
1880 NEXT l
1890 LOCATE i,23-x:PRINT USING "##"; p(j)
1900 NEXT i
1910 GOSUB 2100:GOTO 110
1920 REM
1930 REM Fin du programme
1940 REM
1950 PRINT"Etes-vous sur de bien avoir sauvegarde"
1960 PRINT"les informations comme il convient ?"
1970 PRINT"Si ce n'est pas le cas, entrez"
1980 PRINT"simplement un 'ok' (sinon la touche"
```

```
1990 PRINT"espace suffit)."
2000 INPUT a$
2010 IF a$="ok" THEN CLS:GOTO 920
2020 PRINT:PRINT:PRINT"       Au revoir"
2030 PRINT:PRINT"Maintenant vous pouvez me debrancher"
2040 END
2050 REM
2060 REM SP Attendre
2070 REM
2080 LOCATE 5,25
2090 PRINT"Frappez une touche S.V.P. !"
2100 x$=INKEY$
2110 IF x$="" THEN 2100
2120 CLS:RETURN
```

❐ Liste de variables :

a	réponse au menu.
a$	chaîne de réponse dans la section de programme "fin du travail".
af	valeur initiale de la boucle.
aja	indication de l'année (début).
aja$	tableau d'entrée pour aja.
am	indication du mois (début).
at	indication du jour (début).
dm (i)	montant en francs.
eja	indication de l'année (fin).
eja$	tableau d'entrée pour eja.
em	indication du mois (fin).
et	indication du jour (fin).
ga	nombre de sorties d'argent.
h	variable auxiliaire.
i	index de comptage.
j	index.
ja(i)	année de la sortie d'argent.
ja$(i)	tableau d'entrée pour ja(i).
k(i)	numéro de catégorie pour une sortie d'argent.

k$(i)	désignation de la catégorie.
ka	nombre de catégories définies.
ks(i)	somme des sorties d'argent pour une catégorie donnée dans une période donnée.
l	valeur initiale de la boucle de dessin de l'histogramme.
mo(i)	mois de la sortie d'argent.
n$	nom des informations à sauvegarder.
p(i)	pourcentage des sorties d'argent par catégorie par rapport à l'ensemble des sorties d'argent.
pm	valeur maximale par rapport à la variable p(i).
pp(i)	pourcentage des sorties d'argent par catégorie par rapport à la valeur maximale pm.
su	somme des sorties d'argent pour une période donnée.
tg(i)	jour de la sortie d'argent.
wb$(i)	désignation de la marchandise.
x	variable auxiliaire pour déterminer la valeur finale de la boucle de dessin des histogrammes.
z	valeur arrondie à deux décimales après la virgule pour la variable ks.

❐ Description du programme :

Lignes 10-70 :	Titre.
Lignes 80-90 :	Réservation de place mémoire.
Lignes 100-250 :	Menu et branchement en conséquence.
Lignes 260-600 :	Entrée des sorties d'argent. Dans cette section, deux fenêtres écran sont définies. Alors que les entrées se font dans la première fenêtre (zone gauche de l'écran, définie en ligne 370), les catégories existantes sont sorties dans la seconde fenêtre (zone droite de l'écran, définie en ligne 320). L'utilisateur du programme a ainsi toujours sous les yeux les différentes catégories existantes. Une fois l'entrée terminée, on ne vide en ligne 600 que la première fenêtre.

	Si aucune catégorie n'a encore été définie (voir lignes 290-310) un commentaire est sorti et on saute à la section de programme "Constitution de catégories". L'entrée d'un 0 pour la variable "désignation de la marchandise" a pour effet un retour au menu (ligne 490). Le tableau d'entrée pour la variable ja est une chaîne, ce qui permet en ligne 590 de ne retenir dans le tableau ja(i) que les deux derniers chiffres de l'année (par exemple 86) si vous entrez le nom de l'année avec le siècle (par exemple 1986).
Lignes 610-760 :	Constitution des catégories. Si des catégories ont déjà été définies, celles-ci sont sorties (lignes 640-680). On peut alors définir des catégories. Une fois que 10 catégories ont été créées ou qu'un 0 a été entré, on retourne au menu (voir lignes 730 et 750).
Lignes 770-900 :	Lecture des sorties d'argent et catégories sauvegardées.
Lignes 910-1060 :	Sauvegarde des sorties d'argent et catégories. Lors de la lecture ou de la sauvegarde de variables alphanumériques, il faut noter que cela se produit chaque fois avec une instruction INPUT ou PRINT particulière (voir lignes 830-840 ou lignes 990-1000).
Lignes 1070-1220 :	Examen des sorties d'argent et catégories. Le mode écran est fixé à cet effet sur 80 colonnes en ligne 1100. Une sortie formatée sous forme de tableau est alors réalisée à l'aide de l'instruction PRINT USING. Les variables af et h servent à limiter la sortie à 20 lignes de tableau à la fois. Avant le retour au menu, on revient en mode 40 colonnes (ligne 1220).
Lignes 1230-1910 :	Calculs.

1260-1390 :	Entrée de la période que doivent concerner les calculs. Le texte d'entrée est d'abord sorti, puis les instructions INPUT sont positionnées en conséquence. Les indications d'année sont entrées comme chaînes pour les mêmes raisons qu'en ligne 590.
1400 :	Commentaire.
1410-1420 :	Les variables à calculer su et ks(i) sont fixées sur la valeur initiale 0.
1430-1510 :	Si une sortie d'argent se situe dans la période indiquée, les variables su et ks(i) sont augmentées du montant en francs correspondant. La comparaison des indications d'année, mois et jour pour une sortie d'argent avec la période indiquée ne fonctionne pour ainsi dire que pour notre siècle puisque le siècle n'est pas pris en compte dans l'indication de l'année. L'année 2005 sera par exemple traitée exactement comme l'année 1905.
1520-1650 :	Sortie formatée des montants en francs (par catégorie et montant global) arrondis à deux décimales.
1660-1680 :	Titre pour la représentation graphique de la répartition en pourcentage des différentes catégories de sorties d'argent.
1690 :	La variable pm est fixée à 0.
1700-1730 :	Calcul des pourcentages des sorties d'argent par catégorie par rapport au total des sorties et détermination du pourcentage maximum pm.
1740-1760 :	Calcul des pourcentages des sorties par catégorie par rapport à la valeur maximale de pourcentage pm. Cela sert à optimiser l'utilisation de la place disponible sur l'écran (voir lignes 1850-1880).

1770-1780 :	Dessin d'axes de coordonnées. Avant que les lignes ne soient dessinées avec l'instruction DRAW, le curseur graphique est chaque fois placé sur l'origine des axes avec l'instruction PLOT.
1790-1800 :	Ecriture des axes de coordonnées.
1810 :	La variable d'index j est fixée sur 0.
1820-1900 :	Ecriture de l'axe des x dans cette boucle et dessin des histogrammes. La variable i sert au positionnement horizontal du curseur alors que la variable j identifie les valeurs en pourcentage pp(i) des différentes catégories. La variable x atteint son maximum avec la plus grande valeur en pourcentage, de sorte que les 18 lignes disponibles pour le dessin des histogrammes (lignes 1860-1880) sont utilisées. Pour des valeurs pp(i) plus petites, la valeur finale de la boucle est réduite en proportion (ligne 1860) de sorte que des histogrammes plus courts sont alors dessinés. En ligne 1890, les pourcentages de chaque catégorie sont sortis directement au dessus de l'histogramme correspondant. Comme il s'agit là de valeur arrondies, il se peut que la somme des pourcentages indiqués ne soit pas égale à 100.
1910 :	Fin de cette section de programme.
Lignes 1920-2040 :	Fin du programme. On peut encore, dans cette section de programme, sauvegarder les données entrées si on a oublié de le faire.
Lignes 2050-2120 :	Sous-programme "attendre".

9.5. Traitement de texte

Comparé à des logiciels de traitement de texte professionnels, il est évident que le programme que nous vous proposons dans ce chapitre présente des insuffisances. Mais après tout, il a été conçu uniquement pour un usage domestique et non pour travailler au bureau.

Ce programme vous permet d'effectuer les tâches suivantes :

- Entrée de textes (au clavier).
- Examen de textes (à l'écran).
- Correction de textes (correction, insertion ou suppression de lignes de texte).
- Chargement de textes.
- Sauvegarde de textes.
- Impression de textes.

Vous pouvez choisir vous-même le nombre de caractères par ligne (jusqu'à 70). Vous pouvez également déterminer librement le nombre de lignes par page et la largeur de la marge gauche lors de la sortie sur imprimante. Aucune autre indication de format n'est employée. Le texte est imprimé exactement comme vous l'entrez et comme il apparaît à l'écran.

Contrairement à ce qui est le cas dans notre programme, le formatage, c'est-à-dire la préparation d'un texte pour l'impression, se fait dans beaucoup de programmes de traitement de texte dans une étape de travail particulière. Le moniteur n'est pas en effet en mesure de représenter tout ce que l'imprimante peut sortir. D'autre part, dans beaucoup de programmes de traitement de texte, on n'a pas à surveiller la longueur d'une ligne de texte. Vous pouvez en effet indiquer la longueur de ligne voulue simplement avant la sortie sur imprimante. Vous pouvez naturellement également d'abord faire sortir le texte formaté à l'écran. On peut dire en conclusion que les programmes professionnels vous offrent, outre une manipulation plus aisée du texte, de nombreuses possibilités de sortir sur papier un texte déjà entré en mémoire.

Il en va quelque peu différemment pour le programme que nous vous proposons ici. Le nombre de caractères par ligne est également limité lors de la sortie sur imprimante mais vous devez vous-même veiller lors de l'entrée du texte à ne pas entrer trop de caractères par ligne. Le programme dessine simplement une ligne verticale à l'écran qui vous indique quand la longueur limite d'une ligne est atteinte. C'est le cas chaque fois que le curseur touche cette ligne. Il vous faut alors appuyer sur la touche RETURN ou ENTER avant d'écrire la ligne suivante. Pendant l'entrée du texte, un affichage vous indique sur quelle ligne et dans quelle page vous vous trouvez. Il faut par ailleurs noter dans ce programme que les guillemets doivent être remplacés par l'apostrophe pour ne pas perdre la suite de la ligne.

Un texte traité avec le programme que nous vous proposons peut contenir jusqu'à un maximum de 300 lignes de texte (indépendamment du découpage en pages).

Cela représente environ 6 pages de machine à écrire. Si vous modifiez les lignes 180 et 560, vous pouvez même entrer un nombre plus important de lignes de texte. Vous devez simplement tenir compte de la capacité mémoire de votre ordinateur. Vous pouvez à cet effet interrompre le déroulement du programme pour contrôler avec la fonction FRE la place mémoire encore libre ou bien vous pouvez également intégrer l'interrogation de cette fonction dans votre programme pour empêcher une saturation de la mémoire.

Ceci dit, nous vous invitons à essayer d'améliorer ce programme autant que vous le pourrez. Voici quelques propositions :

a) Intégrer une fonction de recherche avec l'aide de la fonction INSTR (voir le chapitre 9).

b) Modification de l'affectation des touches (voir le chapitre 6.3).

c) Possibilité d'affecter des mots souvent utilisés à certaines touches (KEY, KEY DEF).

d) Intégrer un menu de formatage adaptable à chaque imprimante.

e) Limiter une ligne de texte en s'aidant de la fonction POS.

f) Programmation d'une sortie justifiée. La longueur de chaque ligne de texte doit être pour cela examinée à l'aide des fonctions de traitement de chaînes. Le cas échéant, il faut ajouter d'autres espaces là où il y en a déjà un. On peut alors renoncer à limiter la longueur des lignes de texte (texte d'un seul bloc).

g) Amélioration de l'édition du texte. On pourrait par exemple déterminer avec l'instruction VPOS la ligne de texte devant être corrigée. On pourra alors effacer cette ligne directement pour la remplacer par une nouvelle.

h) Utilisation du second bloc de 64 K pour d'autres textes (voir le chapitre 9).

Comme vous le voyez, vous pouvez mettre à profit, dans un tel programme, grand nombre d'instructions présentées et de conseils donnés dans les chapitres précédents. Vous avez ainsi un bon moyen de confirmer et de développer vos connaissances en BASIC.

Si vous voulez, avec la version du programme que nous vous présentons, compléter un texte déjà sauvegardé, vous pouvez le charger en mémoire et y insérer alors de nouvelles lignes de texte. Les indications de format "caractères par ligne" et "lignes par page" doivent cependant être conservées inchangées. Seule l'indication "lignes par page" peut être encore modifiée lors de la sortie sur imprimante.

Il n'est pas possible de charger un texte à la suite d'un texte déjà en mémoire. Le texte en mémoire serait en effet effacé. Par contre on peut compléter un texte chargé en mémoire en n'importe quel endroit de ce texte.

Pour le reste, ce programme est largement auto-explicatif. Le mieux est que vous l'essayiez directement. Et surtout n'éteignez pas votre ordinateur avant d'avoir sauvegardé votre texte.

```
10 REM Traitement de texte
20 MODE 1
30 PRINT"  Programme de traitement de texte"
40 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
50 PRINT TAB(9) "Bernd Kowal, 1985"
60 GOSUB 2480
70 REM
80 REM Caractères français spéciaux
90 REM
100 SYMBOL AFTER 63
110 SYMBOL 64,&60,&30,&78,&C,&7C,&CC,&76,&0
120 SYMBOL 91,&18,&24,&24,&18,&0,&0,&0,&0
130 SYMBOL 92,&0,&0,&3C,&66,&60,&66,&3C,&18
140 SYMBOL 93,&1E,&30,&38,&6C,&38,&18,&F0,&0
150 SYMBOL 123,&C,&18,&3C,&66,&7E,&60,&3C,&0
160 SYMBOL 124,&30,&18,&66,&66,&66,&66,&3F,&0
170 SYMBOL 125,&30,&18,&3C,&66,&7E,&60,&3C,&0
180 DIM t$(300)
190 REM
200 REM Menu
210 REM
220 PRINT TAB(15)"Menu":PRINT
230 PRINT TAB(33)"Entrée":PRINT
240 PRINT
250 PRINT" entrer texte";TAB(36)"1":PRINT
260 PRINT" examiner texte";TAB(36)"2":PRINT
270 PRINT" corriger texte";TAB(36)"3":PRINT
280 PRINT" charger texte";TAB(36)"4":PRINT
290 PRINT" sauvegarder texte";TAB(36)"5":PRINT
300 PRINT" imprimer texte";TAB(36)"6":PRINT
310 PRINT"Fin du travail";TAB(36)"7"
320 PRINT:PRINT:PRINT:INPUT"Votre choix ";a:CLS
330 IF a<1 OR a>7 THEN 220
340 ON a GOSUB 370,640,840,1720,1820,1940,2250
350 MODE 1:GOTO 220
360 REM
370 REM Entrer textes
380 REM
390 IF zz>0 THEN 420
400 INPUT"Combien de colonnes par ligne ";zz:PRINT
410 IF zz<1 OR zz>70 THEN 400
```

```
420 IF zs>0 THEN 460
430 INPUT"Combien de lignes par page ";zs:PRINT
440 PRINT:PRINT"Si vous entrez '##' pour une ligne,"
450 PRINT"vous mettez fin à l'entrée.":GOSUB 2480
460 GOSUB 2360
470 WINDOW #0,1,7+zz,4,25
480 IF s=0 THEN s=1
490 IF z<1 THEN 540
500 k1=s*zs-zs+1:k2=s*zs-zs+z
510 FOR i=k1 TO k2
520 PRINT #0, USING " ##   ";i-s*zs+zs;
530 PRINT #0, t$(i):NEXT i
540 IF z=zs THEN z=0:s=s+1
550 z=z+1:az=az+1
560 IF az=300 THEN PRINT:PRINT"Il n'y a plus de place disponible
en mémoire":RETURN
570 LOCATE #1, zz/2+7,1:PRINT #1, s
580 PRINT #0, USING " ##   ";z;
590 LINE INPUT "",t$(az)
600 IF t$(az)<>"##" THEN 540
610 t$(az)=""
620 z=z-1:az=az-1:WINDOW #0,1,40,1,25:RETURN
630 REM
640 REM Examen des textes
650 REM
660 INPUT"Quelle page ";x
670 IF x=0 THEN 660
680 l=0:GOSUB 2360
690 WINDOW #0,1,7+zz,4,25
700 LOCATE #1, zz/2+7,1:PRINT #1, x
710 k1=x*zs-zs+1:k2=x*zs
720 FOR i=k1 TO k2
730 l=l+1
740 PRINT #0, USING " ##   ";i-x*zs+zs;
750 PRINT #0, t$(i)
760 IF l=22 THEN l=0:GOSUB 2520
770 NEXT i:GOSUB 2520:MODE 1
780 WINDOW #0,1,40,1,25
790 INPUT"Encore une page (o/n) ";a$
800 IF a$="n" THEN RETURN
810 IF a$<>"o" THEN 790
```

```
820 PRINT:PRINT:GOTO 660
830 REM
840 REM Corriger textes
850 REM
860 PRINT TAB(15)"Menu":PRINT:PRINT
870 PRINT TAB(30)"Entree":PRINT:PRINT
880 PRINT"Travail sur des lignes de exte":PRINT:PRINT
890 PRINT" correction";TAB(33)"1":PRINT
900 PRINT" insertion";TAB(33)"2":PRINT
910 PRINT" suppression";TAB(33)"3":PRINT:PRINT
920 PRINT"Fin de la correction";TAB(33)"4":PRINT
930 PRINT:PRINT:INPUT"Votre choix ";a:MODE 2
940 IF a<1 OR a>4 THEN 860
950 IF a<4 THEN INPUT"Quelle page ";x:PRINT
960 ON a GOTO 980,1140,1460,1690
970 REM
980 REM corriger
990 REM
1000 PRINT"Quelle ligne doit"
1010 INPUT"etre modifiée ";y
1020 i=(x-1)*zs+y
1030 PRINT:PRINT"La ligne précédente :"
1040 PRINT t$(i-1);:PRINT TAB(zz+1) CHR$(211):PRINT
1050 PRINT"La ligne à corriger :"
1060 PRINT t$(i);:PRINT TAB(zz+1) CHR$(211):PRINT
1070 PRINT"La ligne suivante :"
1080 PRINT t$(i+1);:PRINT TAB(zz+1) CHR$(211):PRINT
1090 PRINT:PRINT"Comment doit se présenter la ligne";y;"?"
1100 LOCATE zz+1,17:PRINT CHR$(211):PRINT
1110 LOCATE 1,17:LINE INPUT t$(i)
1120 MODE 1:GOTO 860
1130 REM
1140 REM insertion
1150 REM
1160 PRINT"Entre quelles lignes faut-il placer"
1170 PRINT"l'insertion ?"
1180 PRINT:INPUT"Numéro de ligne inférieur ";y1
1190 z1=y1:y1=(x-1)*zs+y1
1200 PRINT"La ligne se présente ainsi :"
1210 PRINT t$(y1);:PRINT TAB(zz+1) CHR$(211)
1220 PRINT:INPUT"Numéro de ligne supérieur ";y2
```

```
1230 IF y2-z1=1 THEN 1260
1240 PRINT:PRINT"Vous devez indiquer deux numéros de"
1250 PRINT"ligne consécutifs.":GOTO 1160
1260 y2=(x-1)*zs+y2
1270 PRINT"La ligne se présente ainsi :"
1280 PRINT t$(y2);:PRINT TAB(zz+1) CHR$(211)
1290 PRINT:PRINT:PRINT"Combien de lignes faut-il"
1300 INPUT"insérer ";y3
1310 az=az+y3:z=z+y3
1320 IF z>zs THEN z=z-zs:s=s+1
1330 FOR i=az TO y2 STEP -1
1340 t$(i+y3)=t$(i)
1350 NEXT i:GOSUB 2360
1360 WINDOW #0,1,7+zz,4,25
1370 FOR i=y1+1 TO y1+y3
1380 z1=z1+1
1390 LOCATE #1, zz/2+7,1:PRINT #1, x
1400 PRINT #0, USING " ##   ";z1;
1410 LINE INPUT t$(i)
1420 NEXT i
1430 WINDOW #0,1,40,1,25
1440 MODE 1:GOTO 860
1450 REM
1460 REM supprimer
1470 REM
1480 PRINT"Quelles lignes faut-il supprimer?"
1490 PRINT:INPUT"de: numéro de ligne ";y1
1500 z1=y1:y1=(x-1)*zs+y1
1510 PRINT"La ligne se présente ainsi :"
1520 PRINT t$(y1);:PRINT TAB(zz+1) CHR$(211)
1530 PRINT:INPUT"jusqu'à: numéro de ligne ";y2
1540 IF y2>=z1 THEN 1570
1550 PRINT:PRINT"Le second numéro de ligne doit etre"
1560 PRINT"plus grand que le premier.":GOTO 1490
1570 y2=(x-1)*zs+y2
1580 PRINT"La ligne se présente ainsi :"
1590 PRINT t$(y2);:PRINT TAB(zz+1) CHR$(211)
1600 PRINT:PRINT:PRINT TAB(10)"Je supprime !"
1610 GOSUB 2480
1620 y3=y2-y1+1
1630 FOR i=y1 TO az
```

```
1640 t$(i)=t$(i+y3)
1650 NEXT i
1660 az=az-y3:z=z-y3
1670 IF z<1 THEN z=z+zs:s=s-1
1680 MODE 1:GOTO 860
1690 REM Fin de la correction
1700 MODE 1:RETURN
1710 REM
1720 REM Charger textes
1730 REM
1740 INPUT"Nom du texte ";n$:OPENIN n$
1750 INPUT #9,zz,zs,s,z,az
1760 FOR i=1 TO az
1770 LINE INPUT #9,t$(i)
1780 NEXT i
1790 CLOSEIN
1800 CLS:RETURN
1810 REM
1820 REM Sauvegarder textes
1830 REM
1840 PRINT"Comment doit s'appeler le texte"
1850 INPUT"à sauvegarder ";n$:PRINT
1860 OPENOUT n$
1870 PRINT #9,zz,zs,s,z,az
1880 FOR i=1 TO az
1890 PRINT #9,t$(i)
1900 NEXT i
1910 CLOSEOUT
1920 CLS:RETURN
1930 REM
1940 REM Imprimer des textes
1950 REM
1960 PRINT"Faut-il modifier le nombre de lignes"
1970 PRINT"par page (o/n) ";
1980 INPUT a$
1990 IF a$="n" THEN 2020
2000 IF a$<>"o" THEN 1960
2010 PRINT:INPUT"Nouvelle valeur";zs
2020 PRINT:PRINT"Largeur de la marge gauche ?"
2030 INPUT"Nombre d'espaces ";lr
2040 WIDTH zz+lr
```

```
2050 PRINT:PRINT"Faut-il imprimer la totalité"
2060 INPUT"du texte (o/n) ";a$
2070 IF a$="n" THEN 2130
2080 IF a$<>"o" THEN 2050
2090 FOR i=1 TO az
2100 PRINT #8, TAB(lr+1) t$(i)
2110 NEXT i
2120 GOTO 2230
2130 PRINT:PRINT"Quelle page faut-il"
2140 INPUT"imprimer ";x
2150 k1=x*zs-zs+1:k2=x*zs
2160 FOR i=k1 TO k2
2170 PRINT #8, TAB(lr+1) t$(i)
2180 NEXT i
2190 PRINT:PRINT"Faut-il imprimer une autre page"
2200 INPUT"de texte (o/n) ";a$
2210 IF a$="o" THEN 2130
2220 IF a$<>"n" THEN 2190
2230 CLS:RETURN
2240 REM
2250 REM Fin du travail
2260 REM
2270 PRINT"N'avez-vous pas oublié de sauvegarder"
2280 PRINT"le texte ?"
2290 PRINT"Si c'est le cas, entrez simplement"
2300 PRINT"un 'ok'. Sinon, il suffit d'appuyer"
2310 PRINT"sur la touche espace.":INPUT a$
2320 IF a$="ok" THEN GOSUB 1820
2330 PRINT:PRINT:PRINT
2340 PRINT"Vous pouvez maintenant me débrancher."
2350 PRINT:PRINT:PRINT TAB(8)"Au revoir":END
2360 REM
2370 REM Sous-programme masque écran
2380 REM
2390 MODE 2
2400 WINDOW #1,1,80,1,3
2410 PRINT #1,TAB(zz/2)"Page"
2420 PRINT #1,"Ligne";
2430 WINDOW #2,8+zz,9+zz,4,25
2440 FOR i=1 TO 25
2450 PRINT #2, CHR$(211):NEXT i
```

```
2460 RETURN
2470 REM
2480 REM SP Attendre
2490 REM
2500 LOCATE 5,25
2510 PRINT"Frappez une touche S.V.P. !"
2520 x$=INKEY$
2530 IF x$="" THEN 2520
2540 CLS:RETURN
```

❐ Liste de variables :

a	réponse au menu.
a$	chaîne de réponse.
az	nombre de lignes absolu ou numéro de ligne (se rapportant à toutes les lignes de texte figurant en mémoire).
i	index de comptage et numéro de ligne absolu dans la section de programme "correction".
k1	limite inférieure d'une boucle de programme.
k2	limite supérieure d'une boucle de programme.
l	grandeur auxiliaire dans la section de programme "examen du texte".
lr	largeur de la marge gauche lors de la sortie sur imprimante.
n$	nom du texte sauvegardé.
s	nombre de pages.
t$(i)	ligne de texte.
x	nombre de pages dans les sections de programme "examen" et "correction".
y	numéro d'une ligne à corriger.
y1	plus petit numéro de ligne dans les sections de programme "insertion" et "suppression".
y2	plus grand numéro de ligne dans les sections de programme "insertion" et "suppression".
y3	nombre de lignes à insérer ou nombre de lignes à supprimer.
z	nombre de lignes d'une page.
zl	nombre de lignes d'une page dans la section de programme "correction".
zs	lignes par page.
zz	caractères par ligne.

Description du programme :

Lignes 10-60 :	Titre.
Lignes 70-170 :	Définition des caractères spéciaux français.
Ligne 180 :	Réservation de place mémoire pour le texte.
Lignes 190-340 :	Menu de sélection et sauts en conséquence aux différents sous-programmes.
Ligne 350 :	Comme le mode écran est souvent modifié dans les sous-programmes, c'est ce qui est fait ici aussi pour garantir une sortie toujours identique du menu.
Lignes 360-620 :	Sous-programme "entrée du texte".
360-380 :	Commentaire.
390-450 :	Affectation de valeurs aux paramètres zz et zs si cela n'a pas encore été fait et sortie d'une explication pour terminer l'entrée.
460 :	Saut au sous-programme "masque écran".
470 :	Définition d'une zone de l'écran réservée à l'entrée de texte (la limite droite de la fenêtre dépend du paramètre "caractères par ligne").
480 :	La variable s est fixée sur 1 lors de l'entrée de la toute première ligne (s était auparavant égal à 0).
490-530 :	Sortie du texte d'une page partiellement écrite.
540 :	Si une page est pleine (alors z=zs) on passe à une autre page (s=s+1) et la variable z est fixée égale à 0.
550 :	Les variables z et az sont augmentées de 1 avant toute entrée de ligne.

560 :	Sortie d'un commentaire et retour au menu si 300 lignes de texte figurent dans la mémoire programme.
570-580 :	Sortie du nombre de pages et du numéro de ligne (par rapport à une page).
590 :	Entrée d'une ligne de texte. L'index d'une ligne de texte est toujours le numéro de ligne absolu az.
600-620 :	Les variables z et az sont diminuées de 1 lors de l'entrée du critère d'interruption (le critère d'interruption ne fait pas en effet partie du texte proprement dit), la zone écran est redéfinie et on quitte ce sous-programme. Par ailleurs on saute à la ligne 540 pour la poursuite de l'entrée du texte.
Lignes 630-820 :	Sous-programme "examen du texte".
630-650 :	Commentaire.
660-670 :	Entrée d'un nombre de pages.
680-690 :	La variable auxiliaire l est fixée à 0, on saute au sous-programme "masque écran" et la fenêtre de texte est redéfinie.
700 :	Sortie du nombre de pages.
710 :	Les numéros de ligne (absolus) des début et fin d'une page sont calculés.
720-770 :	Sortie du texte. Après sortie de 22 lignes (voir variable auxiliaire l) on saute au sous-programme "attendre". On renonce ici au message "Frappez une touche S.V.P.".
780-820 :	La fenêtre de texte est redéfinie et on peut examiner d'autres pages. Par ailleurs, retour du sous-programme.

Lignes 830-1700 :	Sous-programme "correction du texte".
830-850 :	Commentaire.
860-960 :	Menu et saut correspondant ainsi que demande d'un nombre de pages.
970-1120 :	Section de programme "correction". Un numéro de ligne est réclamé (lignes 1000-1010), le numéro de ligne absolu correspondant est calculé (1020), la ligne à corriger est sortie avec les lignes voisines (1030-1080) puis corrigée (1090-1110). L'écran est enfin vidé et on retourne au menu.
1130-1440 :	Section de programme "insertion". Les numéros de ligne sont réclamés, les nombres caractéristiques absolus sont calculés et les lignes de texte sont sorties (1160-1280). Une valeur est alors entrée pour le nombre de lignes à insérer (1290-1300), les variables z et az sont rectifiées (1310-1320, en tenant compte en ligne 1320 du fait que les lignes supplémentaires nécessitent peut-être une nouvelle page, les indices des lignes de texte existantes sont rectifiés (1330-1350) et les nouvelles lignes de texte peuvent être insérées grâce à la définition de la fenêtre écran et au sous-programme "masque écran" (1350-1430). On revient ensuite au menu (1440).
1450-1680 :	Section de programme "suppression". Les numéros de ligne sont réclamés, convertis et les lignes correspondantes sont sorties (1480-1590). On calcule ensuite (1620), après une sortie de commentaire (1600), le nombre de lignes à supprimer (1630-1650). Les variables z et az sont alors à nouveau calculées (1660) en tenant compte ici aussi d'un changement éventuel du nombre de pages (1670). Enfin, retour au menu (1680).
1690-1700 :	Fin des corrections : vidage de l'écran et retour du sous-programme.

Lignes 1710-1800 :	Sous-programme "chargement d'un texte".
Lignes 1810-1920 :	Sous-programme "sauvegarde de textes". La sauvegarde aussi bien que le chargement s'effectuent exactement comme dans la gestion de fichier du chapitre 9. Le nom d'un texte ne doit bien sûr pas comprendre plus de 8 caractères. Sinon un message d'erreur sera sorti et vous devrez rentrer dans le programme avec un GOTO si vous ne voulez pas perdre le texte en mémoire.
Lignes 1930-2230 :	Sous-programme "Impression du texte". Après quelques indications de format (1960-2030), le nombre de caractères par ligne est limité (2040) et le texte est imprimé en totalité ou page par page. La ligne 2150 est identique à la ligne 710. Dans cette section de programme peuvent encore être ajoutées des instructions supplémentaires pour la commande de l'imprimante (menu d'impression). L'instruction TAB pose notamment des problèmes pour les textes importants. Il vaut mieux alors utiliser des instructions spécifiques pour l'imprimante telles que CHR$(27);'l';CHR$(n); (imprimante Epson), où "n" indique l'emplacement en colonnes de la sortie.
Lignes 2240-2350 :	Sous-programme "fin du travail". Si vous avez oublié de sauvegarder votre texte, vous pouvez encore le faire ici. Sinon, le programme est terminé.
Lignes 2360-2460 :	Sous-programme "masque écran". Le masque est créé en définissant deux fenêtres écran. La première fenêtre (définie en ligne 2400) sert à la sortie des nombres de lignes et de pages actuels. La limite d'une ligne est dessinée (2440-2450) dans la seconde fenêtre (définie en ligne 2430). La définition des fenêtres de texte présente l'avantage de conserver le "masque" même si les lignes de texte disparaissent de l'écran (zone écran numéro 0).
Lignes 2470-2540 :.	Sous-programme "attendre".

9.6. Puissance quatre

Ce livre ne serait pas complet s'il ne vous proposait pas aussi un programme de jeu. Ce n'est pas un hasard si nous avons placé ce programme tout à la fin de l'ouvrage. Il s'agit un peu d'une 'récompense' pour tous les 'efforts' que vous avez fournis dans les chapitres précédents.

Que faites-vous si vous voulez jouer à un jeu pour deux personnes ? La réponse est simple : vous cherchez un partenaire de jeu. Mais que faites-vous si personne ne veut jouer avec vous ? Il n'y a de solution simple à cette question que pour ceux qui possèdent un ordinateur : vous chargez un programme et l'ordinateur remplacera le partenaire de jeu manquant.

Mais comment peut-on arriver à ce que l'ordinateur "réfléchisse", comme il convient pour un partenaire de jeu ? Il vous faut d'abord lui expliquer le terrain de jeu et les règles du jeu. Dans "Puissance quatre", le terrain se compose d'une grille de 8 cases sur 6. Le but du jeu est de constituer une ligne de quatre jetons (horizontale, verticale ou en diagonale). Vos jetons doivent bien sûr être distingués de ceux de votre partenaire de jeu.

Pour obtenir une force moyenne de jeu, il suffit de faire effectuer par votre ordinateur les réflexions suivantes :

❶ Puis-je (moi, l'ordinateur) gagner en un coup ?

❷ Puis-je (moi, l'ordinateur) au coup suivant empêcher que mon adversaire gagne la partie ?

Il est en outre possible d'intégrer des variantes de tactique de jeu dans le déroulement du programme. Pour "Puissance quatre", une tactique, qui mène à la victoire, pourrait consister à essayer de constituer une ligne de trois jetons qui ne soit limitée d'aucun côté par des jetons adverses. Dans les différentes sections du programme, il faudrait alors intégrer des instructions supplémentaires qui commenceraient par

```
IF h=2 AND ... THEN
```

Dans le programme suivant, nous nous limitons toutefois aux deux premières étapes de réflexion indiquées. Si aucune des deux propositions n'est vérifiée, l'ordinateur ne se livre pas à une réflexion tactique mais fait jouer le générateur de hasard. On ne tient donc pas compte des effets du coup qui viendra après le prochain coup.

❐ Un mot concernant la technique de programmation :

L'ordinateur réfléchit en couvrant toute la surface de jeu, c'est-à-dire qu'il peut arriver que soient examinées un très grand nombre de cases qui n'ont encore aucune importance pour le jeu. Une autre technique de programmation, qui réduirait considérablement les temps de calcul dans la phase initiale du jeu, consisterait à limiter la réflexion aux cases occupées. Les différentes instructions IF ... THEN resteraient inchangées dans leur principe.

Il ne nous reste plus qu'à vous souhaiter beaucoup de réussite dans le combat homme contre machine. Faites attention : votre ordinateur ne manquera jamais de remarquer si trois jetons figurent sur une ligne...

```
10 REM Puissance quatre
20 MODE 1
30 PRINT:PRINT TAB(12)"PUISSANCE QUATRE"
40 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
50 PRINT TAB(10) "Bernd Kowal, 1985"
60 GOSUB 1180
70 PRINT"Regles du jeu :":PRINT:PRINT
80 PRINT"Le but du jeu est de placer quatre"
90 PRINT"jetons sur une ligne (peu importe que"
100 PRINT"ce soit horizontalement, verticalement"
105 PRINT"ou en diagonale)."
110 PRINT"Le terrain de jeu est une grille de 8"
120 PRINT"cases sur 6. Les lignes sont numerotees"
130 PRINT"de 1 a 8. Si vous voulez placer par"
140 PRINT"exemple un jeton sur la deuxieme ligne,"
150 PRINT"vous devez entrer un 2. Le jeton"
160 PRINT"tombera jusqu'à ce qu'il tombe sur un"
170 PRINT"autre jeton ou dans la derniere ligne."
```

```
180 PRINT
190 PRINT"Voici mes jetons        : ";CHR$(143);CHR$(143);
CHR$(143)
200 PRINT TAB(27) CHR$(143);CHR$(143);CHR$(143):PRINT
210 PRINT"et voici vos jetons     : ";CHR$(207);CHR$(207);
CHR$(207)
220 PRINT TAB(27) CHR$(207);CHR$(207);CHR$(207):PRINT
230 PRINT:INPUT"Voulez-vous commencer (o/n) ";a$
240 IF a$<>"n" AND a$<>"o" THEN 230
250 CLS:FOR i=1 TO 7:PRINT"  ";
260 IF i=7 THEN PRINT CHR$(147);:GOTO 280
270 PRINT CHR$(151);
280 FOR j=1 TO 31:PRINT CHR$(154);:NEXT j
290 IF i=7 THEN PRINT CHR$(153):GOTO 310
300 PRINT CHR$(157)
310 PRINT:PRINT:NEXT i
320 j=3
330 FOR i=2 TO 19 STEP 3
340 LOCATE j,i:PRINT CHR$(149)
350 LOCATE j,i+1:PRINT CHR$(149)
360 NEXT i
370 IF j=3 THEN j=35:GOTO 330
380 FOR j=7 TO 31 STEP 4
390 FOR i=1 TO 16 STEP 3
400 LOCATE j,i:PRINT CHR$(159)
410 LOCATE j,i+1:PRINT CHR$(149)
420 LOCATE j,i+2:PRINT CHR$(149)
430 NEXT i:LOCATE j,i:PRINT CHR$(155):NEXT j
440 PRINT"  ";:FOR i=1 TO 8
450 PRINT i;" ";
460 r(i,0)=1:NEXT i
470 IF a$="n" THEN s=ROUND(RND*4+2.5):cd=143:GOSUB
1080:co(s,j)=1: GOSUB 1110
480 LOCATE 3,23:INPUT"Votre choix ";s
490 IF s<1 OR s>8 THEN 480
500 cd=207:GOSUB 1080:sb(s,j)=1:GOSUB 1110
510 cd=143:FOR i=1 TO 8
520 FOR j=6 TO 3 STEP -1
530 IF r(i,j)=0 THEN 600
540 IF p=1 THEN 580
550 IF co(i,j)=0 THEN 610
```

```
560 IF co(i,j-1)=1 AND co(i,j-2)=1 THEN s=i:GOSUB 1080:GOSUB
1110:GOTO 1060
570 GOTO 600
580 IF sb(i,j)=0 THEN 610
590 IF sb(i,j-1)=1 AND sb(i,j-2)=1 THEN
r(s,j1)=0:co(s,j1)=0:GOTO 1000
600 NEXT j
610 NEXT i
620 FOR j=1 TO 6
630 sb1=1:sb2=4:h=0
640 FOR i=sb1 TO sb2
650 IF p=1 THEN 680
660 IF co(i,j)=0 THEN i1=i ELSE h=h+1
670 GOTO 690
680 IF sb(i,j)=0 THEN i1=i ELSE h=h+1
690 NEXT i
700 IF p=0 AND h=3 AND r(i1,j)=0 AND r(i1,j-1)=1 THEN s=i1:GOSUB
1080:GOSUB 1110:GOTO 1060
710 IF p=1 AND h=3 AND r(i1,j)=0 AND r(i1,j-1)=1 THEN r(s,j1)=0:
co(s,j1)=0:GOTO 1000
720 IF sb1<6 THEN h=0:sb1=sb1+1:sb2=sb2+1:GOTO 640
730 NEXT j
740 FOR i=1 TO 5
750 FOR j=1 TO 3
760 ip=i:jp=j:h=0
770 IF p=1 THEN 800
780 IF co(ip,jp)=0 THEN ip1=ip:jp1=jp ELSE h=h+1
790 GOTO 810
800 IF sb(ip,jp)=0 THEN ip1=ip:jp1=jp ELSE h=h+1
810 IF ip<i+3 THEN ip=ip+1:jp=jp+1:GOTO 770
820 IF p=0 AND h=3 AND r(ip1,jp1)=0 AND r(ip1,jp1-1)=1 THEN
s=ip1:GOSUB 1080:GOSUB 1110:GOTO 1060
830 IF p=1 AND h=3 AND r(ip1,jp1)=0 AND r(ip1,jp1-1)=1 THEN
r(s,j1)=0:co(s,j1)=0:GOTO 1000
840 NEXT j
850 NEXT i
860 FOR i=4 TO 8
870 FOR j=1 TO 3
880 ip=i:jp=j:h=0
890 IF p=1 THEN 920
900 IF co(ip,jp)=0 THEN ip1=ip:jp1=jp ELSE h=h+1
```

```
910 GOTO 930
920 IF sb(ip,jp)=0 THEN ip1=ip:jp1=jp ELSE h=h+1
930 IF ip>i-3 THEN ip=ip-1:jp=jp+1:GOTO 890
940 IF p=0 AND h=3 AND r(ip1,jp1)=0 AND r(ip1,jp1-1)=1 THEN
s=ip1:GOSUB 1080:GOSUB 1110:GOTO 1060
950 IF p=1 AND h=3 AND r(ip1,jp1)=0 AND r(ip1,jp1-1)=1 THEN
r(s,j1)=0:co(s,j1)=0:GOTO 1000
960 NEXT j
970 NEXT i
980 IF p=1 THEN p=0:GOSUB 1110:GOTO 480
990 FOR i=1 TO 8:s(i)=0:NEXT i:l=0
1000 l=l+1:IF l>8 THEN 1050
1010 IF l=1 THEN s(l)=ROUND(RND*8+0.5):GOTO 1030
1020 s(l)=s(l-1)+1:IF s(l)=9 THEN s(l)=1
1030 s=s(l):GOSUB 1080:co(s,j)=1:j1=j
1040 p=1:GOTO 510
1050 LOCATE 19,23:PRINT"Vous avez gagne":GOTO 1070
1060 LOCATE 19,23:PRINT"J'ai gagne"
1070 END
1080 FOR j=1 TO 6
1090 IF r(s,j)=0 THEN r(s,j)=1:RETURN
1100 NEXT j:IF cd=207 THEN 480 ELSE 1000
1110 z=7:FOR i=2 TO 17 STEP 3
1120 z=z-1:k=0
1130 IF r(s,z)=1 AND r(s,z+1)=1 THEN RETURN
1140 LOCATE s*3+s,i:PRINT CHR$(cd);CHR$(cd);CHR$(cd)
1150 LOCATE s*3+s,i+1:PRINT CHR$(cd);CHR$(cd);CHR$(cd)
1160 IF r(s,z)=0 AND k=0 THEN PEN 0:k=1:GOTO 1140
1170 PEN 1:NEXT i:RETURN
1180 REM SP Attendre
1190 LOCATE 7,25
1200 PRINT"Frappez une touche S.V.P. !"
1210 x$=INKEY$
1220 IF x$="" THEN 1210
1230 CLS:RETURN
```

❒ Liste de variables :

a$ =	chaîne de caractères de réponse (o/n).
cd =	dessin du jeton, expression numérique pour le dessin des jetons dans le sous-programme en lignes 1110-1170.
co(i,j)=	tableau de la grille (i pour la colonne et j pour la ligne); 1 pour occupé et 0 pour libre.
h =	variable de comptage pour le nombre de jetons sur une ligne.
i =	index de comptage.
il =	index de colonne pour une colonne non occupée.
ip =	index de colonne dans l'examen des diagonales.
ip1 =	index de colonne dans l'examen des diagonales pour une case non occupée.
j =	index de comptage.
jp=	index de ligne dans l'examen des diagonales.
jp1 =	index de ligne dans l'examen des diagonales pour une case non occupée.
k =	variable auxiliaire dans le sous-programme en lignes 1110 1170.
l =	index de comptage pour la valeur de la colonne à tester.
p =	variable auxiliaire qui indique laquelle des étapes de réflexion est en train d'être exécutée.
r(i,j) =	tableau de la grille (i pour la colonne et j pour la ligne); 1 pour occupé et 0 pour libre.
s,s(i) =	valeur de colonne choisie ou calculée ou déterminée au hasard.
sb(i,j)=	tableau de la grille (i pour la colonne et j pour la ligne); 1 pour occupé et 0 pour libre.
sb1 =	limite inférieure d'une boucle de programme.
sb2 =	limite supérieure d'une boucle de programme.
z =	index de ligne dans le sous-programme en lignes 1110-1170.

❐ Description du programme :

Lignes 10-60 :	Titre.
Lignes 70-240 :	Sortie des règles du jeu et questions concernant le début du jeu.
Lignes 250-460 :	Le terrain de jeu est dessiné. Les cases de la grille de ligne 0 sont en outre fixées sur 1 pour indiquer à l'ordinateur qu'un jeton ne peut être placé plus bas que la première ligne.
Ligne 470 :	Un nombre aléatoire entre 3 et 6 est formé si l'ordinateur doit commencer le jeu. Les cases correspondantes sont en outre occupées et le jeton est dessiné.
Lignes 480-500 :	Choix de la colonne par l'utilisateur du programme, occupation de la case et dessin du jeton.
Lignes 510-1040 :	L'ordinateur réfléchit. Cette section de programme est d'abord parcourue avec p=0, c'est-à-dire que l'ordinateur examine s'il peut mettre fin au jeu en gagnant. Si ce n'est pas le cas, un parcours est effectué avec p=1, c'est-à-dire que l'ordinateur teste si le nombre aléatoire formé en ligne 1010 ne pourrait pas avoir pour conséquence de faire gagner l'adversaire. Ce nombre aléatoire n'est utilisé par l'ordinateur pour jouer (ligne 980) que si pendant le parcours on n'a pas sauté à la ligne 1000 (test d'un nouveau nombre). On effectue toujours un saut à la ligne 1000 lorsque le choix d'un nombre ou d'une colonne entraînerait la défaite de l'ordinateur. Un tel saut entraîne en même temps une remise à zéro des cases r(s,j1) et co(s,j1) occupées en ligne 1030 pour effectuer les tests.

	Au maximum, les 8 colonnes possibles seront essayées (lignes 1000-1020, seul le premier nombre est tiré au hasard). Après 8 parcours sans succès, l'ordinateur abandonne et saute à la fin du programme (avec sortie d'un commentaire en ligne 1050), c'est-à-dire que l'utilisateur du programme devra parfois se demander pourquoi il a gagné. On peut distinguer les sections de programme suivantes :
510-610 :	Examen des verticales.
620-730 :	Examen des horizontales.
740-850 :	Examen des diagonales allant d'en bas à gauche à en haut à droite.
860-970 :	Examen des diagonales allant d'en bas à droite à en haut à gauche.
980-1040 :	Dessin d'un jeton et saut au choix de l'adversaire si un parcours a été effectué avec succès; formation de nombres de test, occupation-test des cases r(s,j1) et co(s,j1) et retour à la ligne 510.
Lignes 1050-1060 :	Commentaire sur le résultat du jeu.
Ligne 1070 :	Fin du programme.
Lignes 1080-1100 :	Sous-programme pour occuper la case r(s,j) en fonction de la colonne s sélectionnée. Si 6 jetons figurent déjà dans la colonne, un retour est effectué pour un nouveau choix (la variable cd indiquant si c'est l'ordinateur ou l'utilisateur du jeu qui a fait ce choix).
Lignes 1110-1170 :	Sous-programme de dessin d'un jeton en train de tomber.
Lignes 1180-1230 :	Sous-programme "attendre".

Achevé d'imprimer
sur les presses de l'imprimerie IBP
à Rungis (Val-de-Marne 94) (1) 46.86.73.54
Dépôt légal - Novembre 1990
N° d'impression: 5409



LE LIVRE DU BASIC

KOWAL

Grâce à cet ouvrage pratique, maîtrisez les bases de la programmation avec le Basic Locomotive 1.1. Tous les domaines d'applications de ce langage sont traités : graphisme, son, gestion du lecteur de disquettes... Vous découvrirez de nombreux conseils, une foule de trucs et d'astuces destinés à rendre vos applications encore plus performantes. Enfin, disposez de plusieurs programmes complets accompagnés de leur listing : un traitement de texte, un budget familial...

Principaux sujets traités :

- Eléments de base du Basic : variables, arithmétique, fonctions logiques, boucles, branchements, sous-programmes...
- Accès au hardware en Basic : codes ASCII, bits et octets, mémoire...
- Programmation avancée : traitement des chaînes de caractères, tri, gestion du clavier, caractères définis par l'utilisateur...
- Le graphisme : résolution, instructions graphiques, gestion des couleurs, histogrammes et graphisme en 3 dimensions...
- La musique : un éditeur musical, modification des courbes d'enveloppe et de hauteur des notes...
- Le lecteur de disquettes : stockage séquentiel, fonctions d'ouverture et d'écriture de fichiers, fichiers relatifs avec AMSDOS...
- Programmes prêts à l'emploi : budget personnel, remboursement d'un emprunt...



Réf. ML 772. Prix : 99 F.
ISBN : 2-86899-421-0 / ISSN : 0980-1928

EDITIONS MICRO APPLICATION
58 RUE DU FAUBOURG POISSONNIERE
75010 PARIS TEL (1) 47 70 32 44


Document restauré et
retravaillé à partir d'un scan
de mauvaise qualité.
AMSTRAD
CPC
MÉMOIRE ÉCRITE
ACME
https://acpc.me